Provably Efficient Resource Allocation for Edge Service Entities using Hermes

Sheng Zhang, *Member, IEEE,* Yu Liang, Jidong Ge, *Member, IEEE,* Mingjun Xiao, *Member, IEEE,* and Jie Wu, *Fellow, IEEE*

Abstract—Virtualization techniques help edge environments separate the role of the traditional edge providers into two: edge infrastructure providers (EIPs), who manage the physical edge infrastructure, and edge service providers (ESPs), who aggregate resources (especially, compute resources) from multiple EIPs to place service entities and offer value-added services to end users (EUs). In such an environment, end users submit their data analysis jobs to ESPs; ESPs process the data analysis jobs using their service entities. One fundamental and critical problem for an ESP is to decide how much compute resources to rent from each edge server under the constraint that the total amount of rental resources is no more than a specified budget threshold, so that the average makespan of the data analysis jobs submitted to it is minimized. This Edge Resource Allocation (ERA) problem is proven to be NP-complete by reducing the set cover problem to a special case of it. To design an approximation algorithm for ERA, we perform two transformations on ERA: first, we transform ERA into mERA by replacing minimization with maximization; second, we transform mERA into dmERA by limiting the possible amounts of rental resources to a finite set of values. We find that dmERA has several tractable properties that allow us to design Hermes, a provably efficient algorithm that approximates the optimal allocation. We demonstrate that the gap between Hermes and the optimum in simulations and Android-based testbed experiments are no larger than 4.78% and 12.43%, respectively. Hermes can also output a curve showing the trade-off between the average makespan and the budget threshold, so that an ESP can choose the right balance.

Index Terms—Capacity provisioning, edge computing, resource allocation, service entity

1 Introduction

Mobile devices are becoming increasingly popular and pervasive. They are no longer luxuries but musts: a plethora of people use them for banking, gaming, etc. However, mobile devices are, and will continue to be, resource-poor, since the most sought-after features of a mobile device are light weight and tolerable heat dissipation, not high processor speed and large memory size [1]. Therefore, mobile devices still fall short of running computation-intensive jobs, such as augmented reality [2], interactive gaming [3], and natural language processing [4]. A viable solution to overcome this challenge is to offload mobile workloads to remote clouds [5]. However, delivering mobile workloads to remote datacenters incurs long WAN (Wide Area Network) latency, which is not acceptable in many mobile applications such as augmented reality.

Recent studies have proposed deploying small scale edge servers that are geographically near mobile devices and end users [3], [5], [6], [7]. Previous studies mainly focused on automatic application partition for offloading [8], [9], [10], [11], [12], [13], distributed support for machine learning jobs [14], [15], edge service entity placement [16], [17], etc.

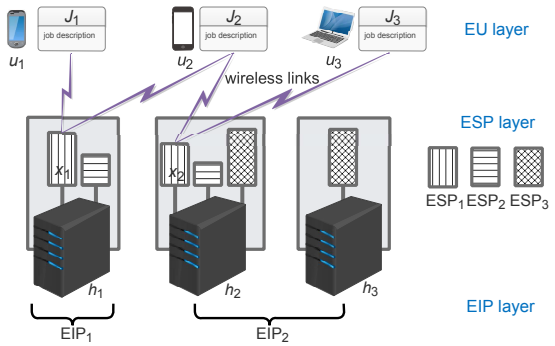


Fig. 1: An example scenario that includes 3 edge servers, 2 EIPs, 3 ESPs, and 3 EUs. Given the constraint that the total amount of rental resources is no more than a specified budget threshold, an ESP (e.g., ESP_1) should decide how many compute resources (e.g., x_1 and x_2) to rent from each edge server so that the average makespan of the data analysis jobs submitted to ESP_1 is minimized.

Edge server resources tend to be virtualized and can be allocated at a fine granularity by the aid of lightweight virtualization techniques [18]. This enables edge virtualization, a paradigm that decouples the functionalities in an edge environment by separating the role of the traditional edge providers into two: edge infrastructure providers (EIPs), who manage the physical edge infrastructure, and edge service providers (ESPs), who aggregate resources (e.g., compute resources) from mul-

- *S. Zhang, Y. Liang, and J.D. Ge are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: sheng@nju.edu.cn, dg1832003@smail.nju.edu.cn, gjd@nju.edu.cn.*
- *M.J. Xiao is with with the School of Computer Science and Technology / Suzhou Institute for Advanced Study, University of Science and Technology of China, Hefei, P. R. China. E-mail: xiaomj@ustc.edu.cn.*
- *J. Wu is with the Center for Networked Computing, Temple University, Philadelphia, PA 19122, USA. E-mail: jiewu@temple.edu.*

tiple EIPs to place service entities and offer value-added services to end users (EUs). In such an environment, end users submit their data analysis jobs to ESPs; ESPs process the data analysis jobs using their service entities. Fig. 1 shows an example. There are 3 edge servers, h_1, h_2, and h_3, owned by 2 EIPs. An ESP may rent compute resources from multiple edge servers, e.g., ESP_1 rents resources from both h_1 and h_2. Three EUs u_1, u_2, and u_3 submit jobs J_1, J_2, and J_3, respectively, to ESP_1 via wireless links. An edge server can offer edge services to an end user if the user is in close proximity of the edge server, e.g., u_1 can only use the resources from h_1, while u_2 can use the resources from both h_1 and h_2. Each end user can process a part of its job locally using a mobile device and offload the rest of its job to the edge servers it can reach.

From the perspective of an ESP, its objective is to minimize the average completion time of the submitted jobs; meanwhile, the ESP wants to minimize the total cost of renting substrate resources from EIPs. Therefore, a fundamental and critical problem for an ESP is to decide how many compute resources to rent from each edge server under the constraint that the total amount of rental resources is no more than a specified budget threshold, so that the average makespan of the data analysis jobs submitted to it is minimized. We call this problem the Edge Resource Allocation (ERA) problem.

The ERA problem is non-trivial due to the following intertwined challenges. First, both of edge servers and user jobs are heterogeneous, so when making the rental decisions, an ESP should respect such heterogeneity. Second, due to the proximity requirement, each edge server can only offer resources to a limited number of users and a user can only use resources from a limited number of edge servers. Third, the goal (i.e., minimizing the average job makespan of a set of jobs) is not linear, making traditional linear or integer linear programming methods no longer effective. These intrinsically intertwined challenges together complicate our problem.

This paper proposes Hermes[1], a provably efficient algorithm that approximates the optimal allocation. The ERA problem is proven to be NP-complete by reducing the set cover problem to a special case of it. To design an approximation algorithm for ERA, we perform two transformations on ERA: first, we transform ERA into mERA by replacing minimization with maximization; second, we transform mERA into dmERA by limiting the possible amounts of rental resources to a finite set of values. We find that dmERA has several tractable properties that allow us to design Hermes. Fortunately, Hermes is also an approximation algorithm for the original ERA problem. We theoretically prove that Hermes is an approximation algorithm for dmERA, mERA, and ERA with approximation factors $\frac{1-1/e}{2G}$, $\frac{1-1/e}{2(n+G)}$, and $\alpha+\beta-\alpha\beta$, respectively, where α and β are defined in Eq. (34). We also discuss three extensions of Hermes. We use simulations and testbed-based experiments to evaluate Hermes and verify our theoretical analysis. We summarize our contributions here as follows:

- We are the first to identify the edge resource allocation problem and prove that ERA is NP-complete, to the best of our knowledge.
- We design Hermes for ERA with guaranteed performance through two-step transformation and two-step reversion. We also discuss extensions and limitations of Hermes.
- We evaluate Hermes using simulations and testbed-based experiments. The simulation results demonstrate that the gap between Hermes and the optimum is 2.21% on average and 4.78% at most. And the testbed results show that the gap between Hermes and the optimum is 12.43% at most.

The rest of the paper is organized as follows. We survey related work in Section 2. We introduce the edge resource allocation problem and its complexity in Section 3. We then present our solution in Section 4. Evaluation is given in Section 5. We conclude the paper and discuss limitations in Section 6.

2 Related Work

There are many works considering efficient offloading for edge computing from both the systemic [10], [11], [12], [13] and algorithmic [8], [9], [19], [20], [21] perspectives. The multi-user computation partition problem with the objective of minimizing the average time is solved in [9]. Dynamic offloading with completion deadline constraint to reduce energy consumption is studied in [8]. Time slot assignment for energy-efficient mobile offloading is investigated in [19]. Tan et al. [20] proposed to greedily dispatch jobs and schedule jobs using the Highest Residual Density First rule, when there are multiple jobs and multiple edge servers. Sundar and Liang [21] investigated the problem of dispatching dependent tasks to multiple edges with deadline constraints, so as to minimize application execution cost. Chen et al. [22] leverages deep reinforcement learning to efficiently dispatch bursty jobs.

Service entity placement was investigated in some recent works. Jia et al. [23] studied the load balancing between multiple edge servers. Yu et al. [16] investigated the problem of joint edge server provisioning and routing path selection from the perspective of networking. Xu et al. [24] considered the caching and offloading problem in resource-limited edge servers to minimize computation latency. Zhang and Tang [25] studied the client assignment problem for DIAs. Liang et al. [26] proposed a utility-based entity placement framework and they also investigated the interaction-oriented edge service entity placement problem [27]. Wang et al. [17] studied a similar service entity placement problem that resembles the uncapacitated facility location problem [28].

1. The name of the algorithm, Hermes, comes from the Olympian God Hermes, who is famous for its unrivalled speed. Similar to the God Hermes, the proposed algorithm minimizes the average job makespan, in other words, it accelerates the job execution speed.

Some other works have discussed the edge support for mobile augmented reality (MAR) applications. Liu et al. [29] focused on edge server assignment and frame resolution selection to minimize MAR service latency. VideoStorm [30] leverages the resource-quality tradeoff and latency-tolerance of partial video analysis requests to accelerate video analysis. Chameleon [31] utilizes temporal persistence of top-k configurations and spatial similarities to minimize the resource consumption for video analysis. JCAB [32] jointly optimizes the configuration of edge-based video analysis and the bandwidth allocation for maximizing the total accuracy and minimizing the total energy cost. These studies explore domain-specific knowledges to optimize edge video analysis.

Some other studies [4], [33], [34], [35], [36] propose pooling together near-by (maybe intermittently-connected) mobile devices for resource sharing, and they form a self-organized cloudlet that collaboratively solves parallel tasks. Workloads are usually assumed fine-grained and permit arbitrary partitioning [37]. How to split workloads during a contact to minimize job makespan is investigated in [34]. How to estimate the computational capacity of a cloudlet is studied in [35].

The power of two choices is due to [28], [38]. Submodular function optimization can be found in [39], [40].

In short, none of existing studies has investigated the resource allocation problem in edge virtualization from the perspective of an edge service provider. We propose solutions with non-trivial performance guarantees. We also reveal the trade-off between makespan and budget.

3 Problem and Complexity

In this section, we first introduce the scenario we consider in this paper, then we present the notations and the problem formulation, lastly we show its time complexity.

3.1 The System

In edge computing environments, edge servers are usually deployed on a business premise such as in a doctor office or a coffee shop [6]. According to the Open Edge Computing initiative [18], edge server resources tend to be virtualized and can be allocated at a fine granularity by the aid of lightweight virtualization techniques. Hence, a mobile device can offload part of its job to nearby edges, i.e., the job is done in parallel by multiple nearby edges. In general, there are two parallelism models: data parallelism and model parallelism [15]. In the first model, the input data is partitioned among the edge servers, and each edge server locally processes the data and returns the results to the mobile device. The second one is usually used in machine learning for training a model which is partitioned among edge servers, and each edge server updates part of the model parameters by processing the entire input data. Data parallelism has been more widely adopted than model parallelism; thus, in this paper, we adopt the data parallelism model.

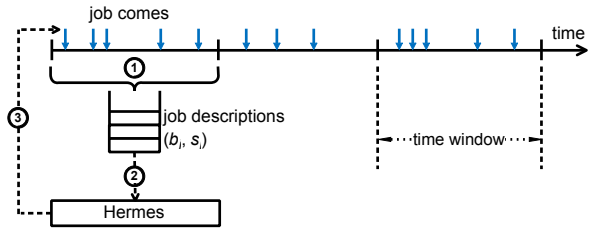


Fig. 2: Scenario overview.

In the data parallelism model, the input data can be classified into two types: modularly divisible and arbitrarily divisible [37]. The workload on the first type of data is usually represented as a directed acyclic graph (DAG) which has dependencies between its small tasks. The workload on the second type of data has the property that all elements in the input demand an identical type of processing [34], [41], [42]. These loads have the characteristic that they can be arbitrarily partitioned into any number of load fractions. Many jobs have this property. e.g., processing of massive experimental data, image processing applications like feature extraction and edge detection, computations of Hough transforms, and extraction of signals buried in noise from multidimensional data collected over large spans of time. Therefore, in this paper, we assume job workloads are fine-grained and permit arbitrary partition. We postpone the discussion on the case in which partition is limited until Section 4.4.

Fig. 2 shows the overview of our scenario. Mobile users[2] submit their jobs in an online manner. Time is divided into multiple time windows of each length. The proposed algorithm, i.e., Hermes, allocates resources to service entities at the end of each time window for jobs submitted within the window. More specifically, when a device u_i submits a job J_i, it would provide the amount of workload of J_i, which is denoted by s_i, and the amount of available local computation resources at u_i. These job descriptions are stored in a queue (see ① in Fig. 2); at the end of each time window, we run Hermes to find how many compute resources to rent from each edge server so that the average makespan of the jobs in the queue is minimized (see ② in Fig. 2); the allocation results are then sent back to each device and edge server (see ③ in Fig. 2).

As mentioned above, we use s_i to denote the amount of workload of J_i, which is quantified by the amount of computations (e.g., the total number of CPU cycles required to accomplish a job is used as the workload of the job in [8]). Without loss of generality, the input size of J_i is assumed to be proportional to the workload size of J_i; thus, we also use s_i to represent the input size of J_i. By offloading partial workloads to some edge servers, the amount of compute resources a job can use consists of two parts: the capacity of the job submitter and the equal-share of each edge server it connects with. Now we have the following question: given the amount of compute resources a job can use, how does the job

2. We will use user and device interchangeably in this paper.

partition its workload to minimize its own makespan?

For example, in Fig. 1, suppose the workload of J_2 is 100, the amount of local compute resources on the submitter u_2 is 10, J_2 can use 10 and 20 units of compute resources in h_1 and h_2, respectively. How can we partition the workload of J_2 to minimize the makespan of J_2? If we send 30 and 40 units of workloads to h_1 and h_2, respectively, the makespan of J_2 is $\max\{\frac{30}{10}, \frac{30}{10}, \frac{40}{20}\} = 3$. It is not hard to see that, to minimize the makespan of J_2, J_2 should finish at the same time on u_2, h_1, and h_2, resulting in the optimal makespan $\max\{\frac{25}{10}, \frac{25}{10}, \frac{50}{20}\} = 2.5$. Therefore, we can calculate the optimal makespan of a single job through dividing the amount of workloads by the total amount of compute resources it can use, e.g., for J_2, its optimal makespan is $\frac{100}{10+10+20} = 2.5$. In the rest of the paper, given the amount of compute resources a job can use, we assume that its makespan is the amount of workloads divided by the total amount of compute resources it can use.

Each device can partition the input of its job based on the allocation results returned by Hermes in the following way. We assume *the compute resources rented by an ESP in each edge server is equally shared among the devices that are in proximity to it.* For example, in Fig. 1, if ESP_1 rents $x_1 = 40$ and $x_2 = 20$ units of compute resources in h_1 and h_2, respectively, then, J_2 can use $\frac{40}{2} = 20$ and $\frac{20}{2} = 10$ units of resources in h_1 and h_2, respectively. Suppose the amount of local computation resources at u_2 is 10 and the input size of J_2 is 160, then the optimal makespan of J_2 is $\frac{160}{20+10+10} = 4$. Therefore, u_2 should send $4 \times 20 = 80$ and $4 \times 10 = 40$ units of input data to h_1 and h_2, respectively, and keep the remaining 40 units of input data for local processing.

As mobile devices are usually in proximity to edges, network latency is usually very small compared with computation latency. Besides, the input data can be partitioned into blocks; when a block is received by a service entity, the entity can start processing it and meanwhile the subsequent blocks are being transmitted. That is, transmitting data and processing data can be done in a pipeline-like way, which further reduces the impact of network latency on the job makespan. Therefore, in this paper, we assume that transmitting a part of a job from a user to an edge server incurs no network latency.

3.2 Problem Formulation

We consider an edge computing scenario which contains a set of n edge servers, denoted by h_1, h_2, ..., and h_n. They are operated by multiple different EIPs. There is one ESP of interest. The budget threshold for the ESP is C. Denote by x_i the amount of compute resources the ESP wants to rent in each edge server. These x_i's are the optimization variables. Obviously, we have

$$\sum\nolimits_{i=1}^{n} x_i \leq C. \tag{1}$$

This paper focuses on delay-sensitive mobile jobs, which demand low delay for improving user experience [9]. Such kind of jobs (e.g., mobile augmented reality) usually perform computation-intensive operations onto the input data and then output the results. There are m data analysis jobs, J_1, J_2, ..., and J_m, submitted by users u_1, u_2, ..., and u_m, respectively. u_i is also called the submitter of J_i. We use u_i to refer to both the user and the mobile device, unless otherwise specified.

The computation workload of a job can be represented in terms of the total number of CPU cycles required for accomplishing the job [8], [43]. In this paper, we let s_i denote the amount of workloads of J_i. To reduce the completion time of a job J_i, the submitter u_i usually offloads partial workloads to nearby edge servers and processes the remaining workloads locally using its own computation resources. Let b_i denote the amount of available computation resources u_i has for its job.

The connections of edge servers and users are represented by a 0-1 matrix $\mathbf{R} = [r_{ij}]_{n \times m}$, where

$$r_{ij} = \begin{cases} 1 & \text{if } u_j \text{ is in proximity to } h_i, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The number of users that are connected to h_i is hence $\sum_{k=1}^{m} r_{ik}$. As we mentioned before, we assume that the compute resources rented by an ESP in each edge server is equally shared among the users that are in proximity to it. In other words, if $r_{ij} = 1$, then J_j obtains $\frac{x_i}{\sum_{k=1}^{m} r_{ik}}$ amount of compute resources (i.e., the clock frequency of the CPU chip [8]) from edge server h_i. Thus, the total amount of compute resources J_j can use is

$$b_j + \sum_{i=1}^{n} \frac{r_{ij} x_i}{\sum\limits_{k=1}^{m} r_{ik}}. \tag{3}$$

As we mentioned in the last subsection, the makespan of a job is the amount of workloads divided by the total amount of compute resources it can use. Therefore, the makespan of J_j can be represented by

$$\frac{s_j}{b_j + \sum\limits_{i=1}^{n} \frac{r_{ij} x_i}{\sum\limits_{k=1}^{m} r_{ik}}}. \tag{4}$$

We use the average makespan, i.e.,

$$\frac{1}{m} \sum_{j=1}^{m} \frac{s_j}{b_j + \sum\limits_{i=1}^{n} \frac{r_{ij} x_i}{\sum\limits_{k=1}^{m} r_{ik}}}, \tag{5}$$

as the optimization goal. The average job makespan indicates, on average, how long it takes for a data analysis job to get its final output. Given a set of jobs, the total number of jobs, i.e., m, is fixed. Thus, minimizing average job makespan is equivalent to minimizing the total makespan. Therefore, the optimization goal can be rewritten as

$$d(\mathbf{X}) = d([x_1, x_2, ..., x_n]) = \sum_{j=1}^{m} \frac{s_j}{b_j + \sum\limits_{i=1}^{n} \frac{r_{ij} x_i}{\sum\limits_{k=1}^{m} r_{ik}}}, \tag{6}$$

TABLE 1: Main notations for quick reference

Symbol	Meaning
m	# of jobs/devices/users
J_i	i-th job
u_i	submitter of J_i
b_i	local computation capacity of u_i
s_i	the amount of workloads of J_i
n	# of edge servers
h_j	j-th edge server
C	the budget threshold for the ESP
r_{ij}	indicating whether u_j is in proximity to h_i
x_i	# of resources the ESP wants to rent in h_i
$d(\mathbf{X})$	the objective function of ERA
$D(\mathbf{X})$	the objective function of mERA
G	# of possible values of x_i in dmERA

where $\mathbf{X} = [x_1, x_2, ..., x_n]$. Main notations are summarized in Table 1 for quick reference.

The ERA problem can be formulated as follows:

$$\begin{aligned} \textsf{[ERA]} \quad \min \quad & d(\mathbf{X}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i \le C \\ & x_i \ge 0, \quad i = 1, 2, ..., n \end{aligned} \tag{7}$$

Note that $d(\mathbf{X})$ is non-linear, therefore, ERA is not a linear programming problem. Taking Fig. 1 for example, suppose there are 3 jobs with $s_1 = 70$, $s_2 = 100$, $s_3 = 50$, $b_1 = 1$, $b_2 = 5$, and $b_3 = 10$. When the budget threshold C for ESP_1 is 90, the optimal solution is $x_1 = 59$ and $x_2 = 31$; when C increases to 180, the optimal allocation changes to $x_1 = 107$ and $x_2 = 73$.

3.3 Complexity

By reducing the NP-complete Set Cover (SC) problem [28] to ERA, we have the following theorem.

Theorem 1: The decision version of ERA is NP-complete.

Proof: We first present the decision versions of SC and ERA as follows.

- *Decision version of SC*: Given a universe $\mathcal{U} = \{e_1, e_2, ..., e_M\}$ of M elements, an integer K, a collection of subsets of $\mathcal{U}$, i.e., $\mathcal{R}_1$, $\mathcal{R}_2$, ..., and $\mathcal{R}_N$, does there exist a sub-collection of these subsets with size no more than K that covers all elements of $\mathcal{U}$?
- *Decision version of ERA*: Given n edge servers, the connection matrix $\mathbf{R}$, the budget threshold C, and m jobs J_1, J_2, ..., and J_m, where J_i has b_j units of local computation resources and s_j units of workloads, is there an assignment of x_i that makes the total makespan no more than D?

We now show that any instance of SC can be polynomially reduced to an instance of ERA. Without loss of generality, denote an arbitrary instance of SC by $< M, N, K, \mathcal{R}_i >$, the corresponding instance of the ERA problem $< m, n, s_j, b_j, r_{ij}, C, D, x_i >$ can be constructed as follows:

- $m \leftarrow M$ and $n \leftarrow N$;
- $s_j \leftarrow M$ and $b_j \leftarrow 1$ for each job;
- $r_{ij} \leftarrow 1$ if $h_j \in \mathcal{R}_i$; otherwise, $r_{ij} \leftarrow 0$;
- $C \leftarrow MK \max\limits_i\{|\mathcal{R}_i|\} + 1$;
- $D \leftarrow M$;
- x_i can be either $\lfloor M \max\limits_i\{|\mathcal{R}_i|\} + \frac{1}{K} \rfloor$ or 0.

It is easy to see that the construction can be finished in polynomial time. Now, it is sufficient to show that these two instances are indeed equivalent.

($\Longleftarrow$) Suppose that ERA has a positive answer, i.e., there is an assignment of x_i such that the total makespan is no more than D. Considering the possible values of each x_i, the ESP can rent $\lfloor M \max\limits_i\{|\mathcal{R}_i|\} + \frac{1}{K} \rfloor$ amount of compute resources from at most K edge servers. Without loss of generality, we assume,

$$x_i = \begin{cases} \lfloor M \max\limits_i\{|\mathcal{R}_i|\} + \frac{1}{K} \rfloor & 1 \le i \le K, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

We now show $\{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_K\}$ is indeed a positive answer to SC, which is equivalent to proving each user connects with at least one of h_1, h_2, ..., and h_K. We prove this by contradiction: suppose some user, say u_j, does not connect with any of h_1, h_2, ..., or h_K, then the makespan of J_j is $s_j/b_j = M = D$. Therefore, the total makespan over all jobs would be larger than D, which contradicts that it is a positive answer to ERA.

($\Longrightarrow$) Suppose that SC has a positive answer. Without loss of generality, we assume $\mathcal{R}_1$, $\mathcal{R}_2$, ..., and $\mathcal{R}_K$ are selected in SC and can cover all elements. For ERA, we let $x_i = \lfloor M \max\limits_i\{|\mathcal{R}_i|\} + \frac{1}{K} \rfloor$ for $1 \le i \le K$, otherwise $x_i = 0$. Firstly, since

$$\sum_{i=1}^{n} x_i \le \lfloor M \max_i\{|\mathcal{R}_i|\} + \frac{1}{K} \rfloor \cdot K \le C, \tag{9}$$

this is a feasible assignment. Note that x_i is not necessarily an integer. Secondly, for any job J_j, since it connects with at least one edge server, its makespan (see Eq. (4)) is no more than

$$\begin{aligned} \frac{s_j}{b_j + \sum\limits_{i=1}^{n} \frac{r_{ij} x_i}{\sum\limits_{k=1}^{m} r_{ik}}} &\le \frac{M}{1 + \frac{\lfloor M \max\limits_i\{|\mathcal{R}_i|\} + \frac{1}{K} \rfloor}{\max\limits_i\{|\mathcal{R}_i|\}}} \\ &< \frac{M}{1+M} < 1. \end{aligned} \tag{10}$$

So the total makesoan of all jobs is no more than $1 \times M = D$, which implies ERA also has a positive answer.

So far we have proved that the decision version of ERA is NP-hard. Given an assignment of x_i's, we can verify whether the total makespan exceeds D in $O(mn)$ time, therefore, ERA belongs to NP. Combining them together, we prove that the decision version of ERA is NP-complete. □

It is nontrivial to directly find an efficient algorithm for ERA. Therefore, in the next section, we first look at some special cases of ERA to reveal the problem structure and find key insights that help us design Hermes for ERA.

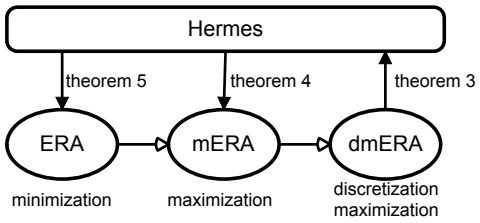


Fig. 3: Theorems 3 to 5 provide theoretical performance bounds of Hermes w.r.t. dmERA, mERA, and ERA, respectively.

4 The Solution: Hermes

Observing that directly solving ERA is not easy, we first perform two transformations on ERA to obtain a new problem dmERA in Section 4.1. Then, we find that dmERA has several tractable properties that allow us to design Hermes in Section 4.2. We provide theoretical analysis on Hermes in Section 4.3. Extensions of Hermes are discussed in Section 4.4.

4.1 Problem Transformation

Given any instance of ERA, $\sum_{j=1}^{m} \frac{s_j}{b_j}$ is fixed. Thus, minimizing $d(\mathbf{X})$ in Eq. (6) is equivalent to maximizing

$$D(\mathbf{X}) = -\sum_{j=1}^{m} \frac{s_j}{b_j + \sum\limits_{i=1}^{n} \frac{r_{ij}x_i}{\sum\limits_{k=1}^{m} r_{ik}}} + \sum\nolimits_{j=1}^{m} \frac{s_j}{b_j}. \quad (11)$$

Then we have the following equivalent problem, where "m" in mERA denotes maximization:

$$\begin{aligned} \text{[mERA]} \quad \max \quad & D(\mathbf{X}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i \le C \\ & x_i \ge 0, \quad i = 1, 2, ..., n \end{aligned} \quad (12)$$

We further transform mERA to dmERA by limiting x_i to a finite set of possible values, where "d" in dmERA denotes discretization:

$$\begin{aligned} \text{[dmERA]} \quad \max \quad & D(\mathbf{X}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i \le C \\ & x_i \in \{0, \frac{C}{G}, ..., \frac{GC}{G}\}, i = 1, 2, ..., n \end{aligned} \quad (13)$$

In dmERA, G is a hyper-parameter: in practice, it could be the number of CPU cores. We can think of G as a knob that we are able to turn and it controls the approximation ratio and running time of Hermes. We will see the impact of G from the perspectives of both theoretical analysis and extensive simulations.

Relationship among ERA, mERA, and dmERA is shown in Fig. 3. The roadmap is outlined as follows. In Section 4.2, we develop an approximation algorithm, i.e., Hermes, for dmERA. In Section 4.3, we prove that Hermes is also an approximation algorithm for both mERA and ERA.

4.2 Solving Discrete Maximization Version of ERA

We start by looking at two special cases of dmERA. Note that, our target is not to solve these two special cases, but to collect useful information for us to solve the general dmERA problem.

4.2.1 Uniform Fixed Rental

In this case, we assume

$$\boxed{\begin{array}{l} x_i \text{ is either 0 or } \frac{FC}{G} \text{ for any edge server } h_i, \\ \text{where } F \text{ is an fixed integer and } 1 \le F \le G. \end{array}}$$

Then, the decision we have to make is to select a subset $\mathcal{H}$ of $\{h_1, h_2, ..., h_n\}$ and make sure that the number of selected edge servers is no more than $\lfloor \frac{C}{\frac{FC}{G}} \rfloor = \lfloor \frac{G}{F} \rfloor$.

The objective function in this case can be rewritten as follows, where "u" denotes uniform:

$$D_u(\mathcal{H}) = -\sum_{j=1}^{m} \frac{s_j}{b_j + \sum\limits_{h_i \in \mathcal{H}} \frac{r_{ij}\frac{FC}{G}}{\sum\limits_{k=1}^{m} r_{ik}}} + \sum_{j=1}^{m} \frac{s_j}{b_j}. \quad (14)$$

The uniform case of dmERA can be formulated as follows:

$$\begin{aligned} \text{[udmERA]} \quad \max \quad & D_u(\mathcal{H}) \\ \text{s.t.} \quad & |\mathcal{H}| \le \lfloor \frac{G}{F} \rfloor \end{aligned} \quad (15)$$

In the following, we prove that $D_u(\mathcal{H})$ has three tractable properties: nonnegativity, monotonicity, and submodularity.

***Definition** 1:* (**Nonnegativity, Monotonicity, and Submodularity**) Given a non-empty finite set $\mathcal{U}$, and a function f defined on the power set $2^{\mathcal{U}}$ of $\mathcal{U}$ with real values, f is called

- *nonnegative* if $f(\mathcal{A}) \ge 0$ for all $\mathcal{A} \subseteq \mathcal{U}$;
- *monotone* if $f(\mathcal{A}) \le f(\mathcal{A}')$ for all $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{U}$;
- *submodular* if $f(\mathcal{A} \cup \{a\}) - f(\mathcal{A}) \ge f(\mathcal{A}' \cup \{a\}) - f(\mathcal{A}')$ for all $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{U}$ and $a \in \mathcal{U} - \mathcal{A}'$.

We have the following theorem:

***Theorem** 2:* $D_u(\mathcal{H})$ in udmERA is nonnegative, monotone, and submodular.

Proof: (Nonnegativity) Since $\sum\limits_{h_i \in \mathcal{H}} \frac{r_{ij}\frac{FC}{G}}{\sum_{k=1}^{m} r_{ik}} \ge 0$ for any J_j, we have

$$\frac{s_j}{b_j + \sum\limits_{h_i \in \mathcal{H}} \frac{r_{ij}\frac{FC}{G}}{\sum\limits_{k=1}^{m} r_{ik}}} \le \frac{s_j}{b_j}, \quad (16)$$

which guarantees that $D_u(\mathcal{H})$ is nonnegative.

(Monotonicity) For all $\mathcal{H} \subseteq \mathcal{H}'$, since, for any J_j,

$$\sum_{h_i \in \mathcal{H}} \frac{r_{ij}\frac{FC}{G}}{\sum\limits_{k=1}^{m} r_{ik}} \le \sum_{h_i \in \mathcal{H}'} \frac{r_{ij}\frac{FC}{G}}{\sum\limits_{k=1}^{m} r_{ik}}, \quad (17)$$

we have $D_u(\mathcal{H}) \le D_u(\mathcal{H}')$, which indicates monotonicity.

Algorithm 1 Algorithm for udmERA

Input: the job size s_j and the local computation capacity b_j for each $j \in [1, m]$, the connection indicator r_{ij} for each pair of $i \in [1, n]$ and $j \in [1, m]$, budget threshold C, hyper-parameter G, fixed integer F

Output: $\mathcal{H}$

1: $\mathcal{H} \leftarrow \emptyset$
2: **while** $|\mathcal{H}| < \lfloor \frac{G}{F} \rfloor$ **do**
3: select $h_i \notin \mathcal{H}$ that maximizes $D_u(\mathcal{H} \cup \{h_i\}) - D_u(\mathcal{H})$
4: $\mathcal{H} \leftarrow \mathcal{H} \cup \{h_i\}$
5: **end while**
6: **return** $\mathcal{H}$

(Submodularity) Denote $(b_j + \sum_{h_i \in \mathcal{H}} \frac{r_{ij}\frac{FC}{G}}{\sum_{k=1}^{m} r_{ik}})$ by $f(\mathcal{H}, j)$. For all $\mathcal{H} \subseteq \mathcal{H}'$ and any edge server h, we have

$$
\begin{aligned}
& f(\mathcal{H}, j) \cdot f(\mathcal{H}', j) \leq f(\mathcal{H} \cup \{h\}, j) \cdot f(\mathcal{H}' \cup \{h\}, j) \\
\Leftrightarrow & \frac{f(\mathcal{H}', j) - f(\mathcal{H})}{f(\mathcal{H}, j) \cdot f(\mathcal{H}', j)} \geq \frac{f(\mathcal{H}', j) - f(\mathcal{H})}{f(\mathcal{H} \cup \{h\}, j) \cdot f(\mathcal{H}' \cup \{h\}, j)} \\
\Leftrightarrow & \frac{s_j}{f(\mathcal{H})} - \frac{s_j}{f(\mathcal{H} \cup \{h\}, j)} \geq \frac{s_j}{f(\mathcal{H}', j)} - \frac{s_j}{f(\mathcal{H}' \cup \{h\}, j)} \\
\Leftrightarrow & D_u(\mathcal{H} \cup \{h\}) - D_u(\mathcal{H}) \geq D_u(\mathcal{H}' \cup \{h\}) - D_u(\mathcal{H}'),
\end{aligned}
$$

which indicates that $D_u(\mathcal{H})$ is submodular. □

Theorem 2 enables us to design an approximation algorithm of factor $(1 - 1/e)$ shown in Alg. 1, where e is the base of natural logarithm [40]. Remember that x_i can only be 0 or $\frac{FC}{G}$ in udmERA, we only have to select a subset of edge servers. In Alg. 1, $\mathcal{H}$ is initialized to $\emptyset$; in each iteration, we add the edge server that maximizes the marginal gain of $D_u(\mathcal{H})$ into $\mathcal{H}$, *i.e.*, in each iteration, we select $h_i \notin \mathcal{H}$ that maximizes $D_u(\mathcal{H} \cup \{h_i\}) - D_u(\mathcal{H})$.

There are at most n iterations in Alg. 1; in each iteration, we need to check at most n clouds to find the cloud that maximizes the marginal gain. It takes $O(mn)$ time to compute $D_u(\mathcal{H})$, thus, the time complexity of Alg. 1 is $O(mn^3)$.

4.2.2 Non-uniform Fixed Rental

In this case, we assume

$$
\boxed{\begin{array}{c} x_i \text{ is } \frac{f_i C}{G} \text{ for edge server } h_i, \\ \text{where } f_i \text{ is an fixed integer and } 0 \leq f_i \leq G. \end{array}}
$$

Then, the decision we have to make is also to select a subset $\mathcal{H}$ of $\{h_1, h_2, ..., h_n\}$ and make sure that the total amount of rental resources in selected edge servers is no more than C, i.e., $\sum_{h_i \in \mathcal{H}} x_i = \sum_{h_i \in \mathcal{H}} \frac{f_i C}{G} \leq C$, which is equivalent to $\sum_{h_i \in \mathcal{H}} f_i \leq G$.

The objective function in this case can be rewritten as follows, where "n" denotes non-uniform:

$$
D_n(\mathcal{H}) = -\sum_{j=1}^{m} \frac{s_j}{b_j + \sum\limits_{h_i \in \mathcal{H}} \frac{r_{ij}\frac{f_i C}{G}}{\sum\limits_{k=1}^{m} r_{ik}}} + \sum_{j=1}^{m} \frac{s_j}{b_j}. \tag{18}
$$

Algorithm 2 Algorithm for ndmERA

Input: the job size s_j and the local computation capacity b_j for each $j \in [1, m]$, the connection indicator r_{ij} for each pair of $i \in [1, n]$ and $j \in [1, m]$, budget threshold C, hyper-parameter G, fixed integer f_i for each $i \in [1, n]$

Output: $\mathcal{H}$

1: call Alg. 1 to generate $\mathcal{H}_1$
2: $\mathcal{H}_2 \leftarrow \emptyset$
3: **while** $G \geq \sum\limits_{h_i \in \mathcal{H}_2} f_i + \min\limits_{h_i \notin \mathcal{H}_2} f_i$ **do**
4: select $h_i \notin \mathcal{H}_2$ that maximizes $\frac{D_n(\mathcal{H}_2 \cup \{h_i\}) - D_n(\mathcal{H}_2)}{f_i}$ subject to $\sum\limits_{h_j \in \mathcal{H}_2 \cup \{h_i\}} f_i \leq G$
5: $\mathcal{H}_2 \leftarrow \mathcal{H}_2 \cup \{h_i\}$
6: **end while**
7: **return** $\arg \max\limits_{\mathcal{H}' \in \{\mathcal{H}_1, \mathcal{H}_2\}} D_n(\mathcal{H})$

The non-uniform dmERA can be formulated as follows:

$$
\begin{aligned}
\text{[ndmERA]} \quad & \max \quad D_n(\mathcal{H}) \\
& \text{s.t.} \quad \sum_{h_i \in \mathcal{H}} f_i \leq G
\end{aligned} \tag{19}
$$

To solve ndmERA, an intuitive idea is to use the same greedy heuristic in Alg. 1. Another intuitive idea is that, in each iteration, we select the edge server that maximizes the ratio of marginal gain of $D_n(\mathcal{H})$ to the amount of rental resources in that edge server, i.e., in each iteration, we select $h_i \notin \mathcal{H}$ that maximizes $\frac{D_n(\mathcal{H} \cup \{h_i\}) - D_n(\mathcal{H})}{f_i}$. However, there is no theoretic performance guarantee on either of them. Fortunately, if we use these two ideas independently and return the better one of the two results, then the performance is bounded [38], [40], [44]: the approximation ratio is $\frac{1}{2}(1 - \frac{1}{e})$. The algorithm is shown in Alg. 2. It is not hard to see the time complexity of the algorithm is also $O(mn^3)$.

4.2.3 General dmERA

We design Hermes (shown in Alg. 3) for dmERA. The main intuition behind Hermes is as follows: since we already have an approximation algorithm (Alg. 2) for ndmERA, is it possible for us to find another intermediate problem that is (a) similar to ndmERA, and (b) easy to transform the results to fit for dmERA?

Fortunately, such an intermediate problem exists, and we name it idmERA, where "i" denotes intermediate: there are m jobs with parameters b_j and s_j, n groups of edge servers, each group has G edge servers, the coverages of edge servers in the same group are the same. More specifically, the i-th group of edge servers are denoted by h_{i1}, h_{i2}, ..., and h_{iG}. We use x_{ik} to denote the amount of resources we would like to rent in edge server h_{ik}. Similar to ndmERA, we assume $x_{ik} = \frac{kC}{G}$. The coverage can be represented by a three-dimensional matrix $[r_{ikj}]_{n \times G \times m}$, where $r_{ikj} = 1$ if h_i connects with u_j, otherwise $r_{ikj} = 0$.

The budget threshold is *still* C, the question is how to choose a subset $\mathcal{H}$ of these nG edge servers to maximize

$$Q(\mathcal{H}) = -\sum_{j=1}^{m} \frac{s_j}{b_j + \sum_{i,k} \frac{r_{ikj}x_{ik}}{\sum_{h=1}^{m} r_{ikh}}} + \sum_{j=1}^{m} \frac{s_j}{b_j}. \tag{20}$$

Notice that, idmERA is in fact a large instance of ndmERA, where the number of edge servers becomes nG. Keeping this observation in mind, let us look at Hermes in Alg. 3. Lines 2-10 generate a solution, $[y'_{ig}]_{n\times G}$, for idmERA using the same idea as in Alg. 1. Here, y'_{ig} indicates whether the edge server h_{ig} is selected. Lines 11-13 transform $[y'_{ig}]_{n\times G}$ into $\mathbf{X}'$ that fits for dmERA: for the i-th group of edge servers, we set x'_i to be the maximal amount of rental resources among all selected edge servers in this group and remove the rest (if any). After doing this, we get some unused budget due to removal, which is $(C - \sum_{i=1}^{n} x'_i)$. We then allocate them in a greedy manner: in each iteration, add $\frac{C}{G}$ to the x'_i that maximizes the marginal objection function, finally we get $\mathbf{X}'$. Lines 15-26 construct another solution $\mathbf{X}''$ using the other idea as in Alg. 2. The final result is the better one of $\mathbf{X}'$ and $\mathbf{X}''$.

The time complexity of Hermes is $O(mn^3G^3)$. The theorems in the next subsection indicate that Hermes has performance guarantee. We will shortly see in extensive evaluations and testbed-based experiments that, Hermes is far better than this theoretical bound.

4.3 Theoretic Analysis

***Theorem** 3:* Hermes is a factor $\frac{1-1/e}{2G}$ approximation algorithm for dmERA.

Proof: Denote by $\mathbf{X}^*$ the optimal solution to dmERA, and by $\mathbf{X}$ the solution returned by Hermes. We want to prove

$$\frac{D(\mathbf{X})}{D(\mathbf{X}^*)} \geq \frac{1-1/e}{2G}. \tag{21}$$

Let $\mathcal{H}^*$ be the optimal solution to idmERA, and let $\mathcal{H}'$ be $\arg\max_{\mathcal{H}\in\{\mathcal{H}_1,\mathcal{H}_2\}} Q(\mathcal{H})$. According to previous results [38], [40], we know

$$Q(\mathcal{H}') \geq \frac{1-1/e}{2} Q(\mathcal{H}^*). \tag{22}$$

Notice that, if we can only select one cloud from each cloud group, idmERA is equivalent to dmERA, *i.e.*, dmERA is a special case of idmERA. With this observation, we have

$$Q(\mathcal{H}^*) \geq D(\mathbf{X}^*). \tag{23}$$

Looking at lines 11-13 and 24-26 of Hermes, when transforming $[y'_{ig}]_{n\times G}$ into $\mathbf{X}'$, we set x'_i to be the maximal amount of rental resources among all selected edge servers in this group and remove the rest. Considering there are at most G edge servers in each group, we have

$$\begin{aligned} D(\mathbf{X}') &\geq \frac{Q(\mathcal{H}_1)}{G}, \\ D(\mathbf{X}'') &\geq \frac{Q(\mathcal{H}_2)}{G}. \end{aligned} \tag{24}$$

Algorithm 3 Hermes

Input: the job size s_j and the local computation capacity b_j for each $j \in [1, m]$, the connection indicator r_{ij} for each pair of $i \in [1, n]$ and $j \in [1, m]$, budget threshold C, hyper-parameter G
Output: $\mathbf{X} = [x_1, x_2, ..., x_n]$

1: // construct $\mathbf{X}'$
2: set $x_i \leftarrow 0$ for each $i \in [1, n]$
3: set $y'_{ig} \leftarrow 0$ for each $i \in [1, n]$ and $g \in [1, G]$
4: set $Y \leftarrow 0$
5: **while** $Y \leq G$ **do**
6: select the y'_{ig} that (1) $y'_{ig} = 0$ and (2) maximizes $D([x_1, ..., x_i + y'_{ig} \cdot \frac{C}{G}, ..., x_n]) - D([x_1, ..., x_i, ..., x_n])$
7: set $y'_{ig} \leftarrow 1$
8: set $x_i \leftarrow x_i + y'_{ig} \cdot \frac{C}{G}$
9: set $Y \leftarrow Y + g$
10: **end while**
11: set $y'_{i0} \leftarrow 1$ for each $i \in [1, n]$
12: set $x'_i \leftarrow \frac{C}{G} \cdot \max_{y'_{ig}=1, g\in[0,G]} g$ for each $i \in [1, n]$
13: allocate the remaining budget (i.e., $C - \sum_{i=1}^{n} x'_i$) in a greedy manner: in each iteration, add $\frac{C}{G}$ to the x'_i that maximizes $D([x'_1, ..., x'_i + \frac{C}{G}, ..., x'_n]) - D([x'_1, ..., x'_i, ..., x'_n])$
14: // construct $\mathbf{X}''$
15: set $x_i \leftarrow 0$ for each $i \in [1, n]$
16: set $y''_{ig} \leftarrow 0$ for each $i \in [1, n]$ and $g \in [1, G]$
17: set $Y \leftarrow 0$
18: **while** $Y \leq G$ **do**
19: select the y''_{ig} that (1) $y''_{ig} = 0$ and (2) maximizes $\frac{D([x_1,...,x_i+y''_{ig}\cdot\frac{C}{G},...,x_n]) - D([x_1,...,x_i,...,x_n])}{g}$
20: set $y''_{ig} \leftarrow 1$
21: set $x_i \leftarrow x_i + y''_{ig} \cdot \frac{C}{G}$
22: set $Y \leftarrow Y + g$
23: **end while**
24: set $y''_{i0} \leftarrow 1$ for each $i \in [1, n]$
25: set $x''_i \leftarrow \frac{C}{G} \cdot \max_{y''_{ig}=1, g\in[0,G]} g$ for each $i \in [1, n]$
26: allocate the remaining budget (i.e., $C - \sum_{i=1}^{n} x''_i$) in a greedy manner: in each iteration, add $\frac{C}{G}$ to the x''_i that maximizes $D([x''_1, ..., x''_i + \frac{C}{G}, ..., x''_n]) - D([x''_1, ..., x''_i, ..., x''_n])$
27: // return the better one of $\mathbf{X}'$ and $\mathbf{X}''$
28: **return** $\arg\max_{\mathbf{X}\in\{\mathbf{X}',\mathbf{X}''\}} D(\mathbf{X})$

Combining Eqs. (22), (23), and (24) together, we have

$$\begin{aligned} D(\mathbf{X}) &= \max\{D(\mathbf{X}'), D(\mathbf{X}'')\} \\ &\geq \frac{\max\{Q(\mathcal{H}_1), Q(\mathcal{H}_2)\}}{G} = \frac{Q(\mathcal{H}')}{G} \\ &\geq \frac{1-1/e}{2G} Q(\mathcal{H}^*) \\ &\geq \frac{1-1/e}{2G} D(\mathbf{X}^*). \end{aligned} \tag{25}$$

The theorem holds immediately. □

In the last theorem, we have shown that Hermes is a

factor $\frac{1-1/e}{2G}$ approximation algorithm for dmERA. Remember that dmERA differs from mERA in the possible values of x_i's: the former limits any x_i to one of 0, $\frac{C}{G}$, ..., $\frac{(G-1)C}{G}$, and C, while the latter allows any x_i to be any real number between 0 and C. We show below that, although Hermes is designed for dmERA, it has guaranteed performance for mERA.

Theorem 4: Hermes is a factor $\frac{1-1/e}{2(n+G)}$ approximation algorithm for mERA.

Proof: Remember that, mERA allows x_i's to be assigned any real values between 0 and C.

Denote by $\mathbf{X}^\dagger$ the optimal assignment of x_i's in mERA, by $\mathbf{X}^*$ the optimal assignment in dmERA, and by $\mathbf{X}$ the solution returned by Hermes. We want to know the relation between $\mathbf{X}^\dagger$ and $\mathbf{X}$. Since we already know the relation between $\mathbf{X}^*$ and $\mathbf{X}$ due to Theorem 3, what we are going to do is to establish the relation between $\mathbf{X}^\dagger$ and $\mathbf{X}^*$.

We construct another assignment $\mathbf{X}^\ddagger$ by rounding each $x_i^\dagger$ in $\mathbf{X}^\dagger$ to the nearest multiples of $\frac{C}{G}$ towards infinity, i.e.,

$$x_i^\ddagger = \frac{f_i C}{G}, \tag{26}$$

where f_i satisfies

$$\frac{(f_i-1)C}{G} < x_i^\dagger \leq \frac{f_i C}{G}. \tag{27}$$

Denote the total amount of rental resources in $\mathbf{X}^\ddagger$ by $C^\ddagger$, which is

$$C^\ddagger = C + \sum_{i=1}^{n}(x_i^\ddagger - x_i^\dagger) \leq C + \frac{nC}{G}. \tag{28}$$

Obviously, we have

$$D(\mathbf{X}^\dagger) \leq D(\mathbf{X}^\ddagger)|_{C^\ddagger} \leq D(\mathbf{X}^\ddagger)|_{C+\frac{nC}{G}}, \tag{29}$$

where $D(\mathbf{X}^\ddagger)|_{C^\ddagger}$ represents the objective function when the budget threshold becomes $C^\ddagger$.

Let us take a close look at $\mathbf{X}^\ddagger$. If we take $\frac{C}{G}$ as a unit of resource, $\mathbf{X}^\ddagger$ contains $(n+G)$ units of resources. Consider removing the rental resources in $\mathbf{X}^\ddagger$ unit by unit in a greedy manner: each time we remove the unit that incurs the least marginal loss. After removing n units, we get a new assignment $\mathbf{X}^\S$ that contains exactly G units of resources.

As the removal is conducted in a greedy manner with respect to marginal loss, it is easy to see that, $D(\mathbf{X}^\S)$ is no less than $\frac{G}{G+n}$ percent of $D(\mathbf{X}^\ddagger)|_{C+\frac{nC}{G}}$, that is,

$$D(\mathbf{X}^\S) \geq \frac{G}{G+n} D(\mathbf{X}^\ddagger)|_{C+\frac{nC}{G}}. \tag{30}$$

Observing that $\mathbf{X}^\S$ is a feasible assignment to dmERA with respective to budget C, we have

$$D(\mathbf{X}^\S) \leq D(\mathbf{X}^*). \tag{31}$$

Combining Eqs. (29), (30), and (31) together, we have

$$D(\mathbf{X}^*) \geq \frac{G}{n+G} D(\mathbf{X}^\ddagger)|_{C+\frac{nC}{G}} \geq \frac{G}{n+G} D(\mathbf{X}^\dagger). \tag{32}$$

Taking Eq. (32) and Theorem 3 together, we further have

$$D(\mathbf{X}) \geq \frac{1-1/e}{2G} D(\mathbf{X}^*) \geq \frac{1-1/e}{2(n+G)} D(\mathbf{X}^\dagger). \tag{33}$$

The theorem holds immediately. □

We have shown that Hermes is an approximation algorithm for mERA. Remember that mERA differs from ERA only in the objective function: the former maximizes $D(\mathbf{X})$ in Eq. (11), while the latter minimizes $d(\mathbf{X})$ in Eq. (6). We show below that, Hermes approximates the optimal solution to ERA.

Theorem 5: Hermes is a factor $(\alpha+\beta-\alpha\beta)$ approximation algorithm for ERA, where

$$\alpha = \frac{1-1/e}{2(n+G)}, \text{ and } \beta = \frac{C + \max\limits_j\{b_j\}}{\min\limits_j\{b_j\}}. \tag{34}$$

Proof: According to Eqs. (6), (11), and (33), we have

$$-d(\mathbf{X}) + \sum_{j=1}^{m}\frac{s_j}{b_j} \geq \alpha(-d(\mathbf{X}^\dagger) + \sum_{j=1}^{m}\frac{s_j}{b_j}), \tag{35}$$

which is equivalent to

$$d(\mathbf{X}) \leq \alpha d(\mathbf{X}^\dagger) + (1-\alpha)\sum_{j=1}^{m}\frac{s_j}{b_j}. \tag{36}$$

In the next, we are trying to express $\sum_{j=1}^{m}\frac{s_j}{b_j}$ using $d(\mathbf{X}^\dagger)$. We first develop a lower bound on $d(\mathbf{X}^\dagger)$. Obviously, $d(\mathbf{X}^\dagger)$ is minimized when every application can occupy all the edge server capacities, therefore,

$$d(\mathbf{X}^\dagger) \geq \sum_{j=1}^{m}\frac{s_j}{b_j + C} \geq \frac{\sum\limits_{j=1}^{m} s_j}{C + \max\limits_j\{b_j\}}. \tag{37}$$

Combining Eqs. (36) and (37) together, we have

$$\begin{aligned} d(\mathbf{X}) &\leq \alpha d(\mathbf{X}^\dagger) + (1-\alpha)\sum_{j=1}^{m}\frac{s_j}{b_j} \\ &\leq \alpha d(\mathbf{X}^\dagger) + (1-\alpha)\frac{\sum\limits_{j=1}^{m} s_j}{\min\limits_j\{b_j\}} \\ &\leq \alpha d(\mathbf{X}^\dagger) + (1-\alpha)\frac{(C+\max\limits_j\{b_j\})d(\mathbf{X}^\dagger)}{\min\limits_j\{b_j\}} \\ &\leq (\alpha+\beta-\alpha\beta)d(\mathbf{X}^\dagger). \end{aligned} \tag{38}$$

The theorem holds immediately. □

It should be noted that, the approximation ratio of Hermes in theory may be loose; however, we will shortly see in performance evaluations that, the performance of Hermes is far better than the theoretical bound.

4.4 Discussions

In this section, we discuss several extensions of Hermes to handle various situations including weighted shared, limited partition, and heterogeneous propagation delays.

Weighted Shares. Hermes assumes an edge server is equally-shared among the devices it connects with. In reality, although mobile users may have varied priorities and demands, extending Hermes to fit for this scenario is not hard. We can assign a weight w_j to each user u_j, where w_j may be proportional to the price that u_j pays to the ESP. In this case, $d(\mathbf{X})$ becomes

$$d(\mathbf{X}) = \sum_{j=1}^{m} \frac{s_j}{b_j + \sum_{i=1}^{n} \frac{r_{ij} w_j x_i}{\sum_{k=1}^{m} r_{ik} w_k}}. \tag{39}$$

It is not hard to verify that the new $d(\mathbf{X})$ still has the properties discussed in this paper; thus, we only have to slightly adjust Hermes to fit for weighted shares.

Limited Partition. So far, the Hermes algorithm works only for the arbitrarily divisible workloads, as we mentioned in Section 3.1. This subsection discusses the case in which the partition is limited. We assume the input of any job consists of multiple indivisible blocks. Without loss of generality, the size of a block is denoted as δ. Then, the size s_j of the input of J_j can be denoted by $p_j\delta$, where p_j is the number of blocks. We extend Hermes as follows to solve this case.

We first run Hermes to obtain the resource allocation $\mathbf{X} = [x_1, x_2, ..., x_n]$. Then, we know the total amount of computation resources J_j can use is $b_j + \sum_{i=1}^{n} \frac{r_{ij}x_i}{\sum_{k=1}^{m} r_{ik}}$. Denote this value as η_j. Then, mobile device u_j should send $y_i = \frac{s_j}{\eta_j} \cdot \frac{r_{ij}x_i}{\sum_{k=1}^{m} r_{ik}}$ units of input data to the service entity located at edge server h_i. However, these y_i's may not be exactly a multiple of δ. We then round down these y_i's and let the remaining input data, which is $s_j - \sum_{i=1}^{n} \lfloor y_i \rfloor$, be sent to $\frac{s_j - \sum_{i=1}^{n} \lfloor y_i \rfloor}{\delta}$ randomly selected edge servers that are connected to u_j.

By doing so, at most one more block is sent to an edge server, compared to the divisible case. Suppose the service entity at h_i receives one more block, then the completion time of workloads from J_j at h_i is at most $\frac{\delta r_{ij}}{\frac{x_i}{\sum_{k=1}^{m} r_{ik}}}$ larger than that in the divisible case. Therefore, the makespan of J_j is at most $\frac{\delta r_{ij}}{\min_i\{\frac{x_i}{\sum_{k=1}^{m} r_{ik}}\}}$ longer than that in the divisible case.

Heterogeneous Propagation Delays. We mentioned in Section 3.1 that the network latency is ignored in the proposed algorithm Hermes, since transmitting data and processing data can be simultaneously done in a pipeline-like way. This subsection discusses the case in which the heterogeneous propagation delays between users and edge servers are taken into account.

Suppose the propagation delays between u_j and h_i is L_{ij}. Without loss of generality, we assume $r_{1j} = 1$ and $L_{1j} = \max_{1 \le i \le n}(r_{ij}L_{ij})$. Remember that a job is finished if all parts of the job are finished. To minimize the makespan of J_j, we should let all parts of J_j finish at the same time. Therefore, the makespan of J_j can be denoted by

$$\frac{s_j - b_j L_{1j} - \sum_{i=1}^{n} (L_{1j} - L_{ij}) \frac{r_{ij}x_i}{\sum_{k=1}^{m} r_{ik}}}{b_j + \sum_{i=1}^{n} \frac{r_{ij}x_i}{\sum_{k=1}^{m} r_{ik}}} + L_{1j}. \tag{40}$$

Take J_2 in Fig. 1 for example. Let $s_2 = 100$, $b_2 = 10$, $L_{12} = 0.5$, $L_{22} = 1$, and J_2 can use 10 and 20 units of compute resources in h_1 and h_2, respectively. Then, the optimal makespan of J_2 is $\frac{100-10\times1-(1-0.5)\times10}{10+10+20} + 1 = 3.125$. That is, u_2 sends $(3.125 - L_{12}) \times 10 = 26.25$ and $(3.125 - 1) \times 20 = 42.5$ units of workloads to h_1 and h_2, respectively; and keeps the remaining 31.25 units of workloads for local processing.

Given the makespan of each job, it is easy to verify the new objective still has the properties discussed in this paper, which helps us slightly modify Hermes to adapt to this situation.

5 Performance Evaluation

We evaluate Hermes using simulations, trace-driven and testbed-based experiments. We answer the following questions in our evaluation: (1) How effective is Hermes's resource allocation? (2) How well does Hermes approximate the optimal allocation? (3) What is the effect of the hyper-parameter G? (4) Can Hermes provide any suggestions on choosing the budget threshold?

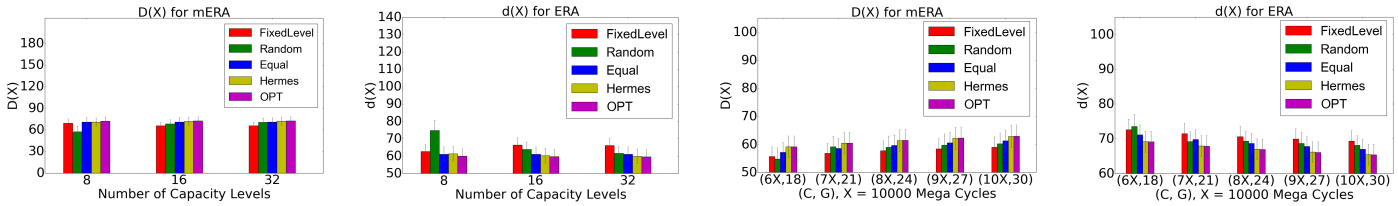
5.1 Simulation-based Evaluation

Our simulations are setup as follows.

Similar to [8], for mobile device capacities, we uniformly generated each b_j between 100 to 1000 Mega cycles per second; the amount of workloads of a job was uniformly generated between 500 to 1500 Mega cycles. For edge server h_i, we let $r_{ij} = 1$ with a probability that is randomly selected between $[1/15, 1/5]$. By default, the number of jobs was 50; the number of edge servers was 6; the budget threshold was 10,000 Mega cycles per second; and the hyper-parameter G was 20.

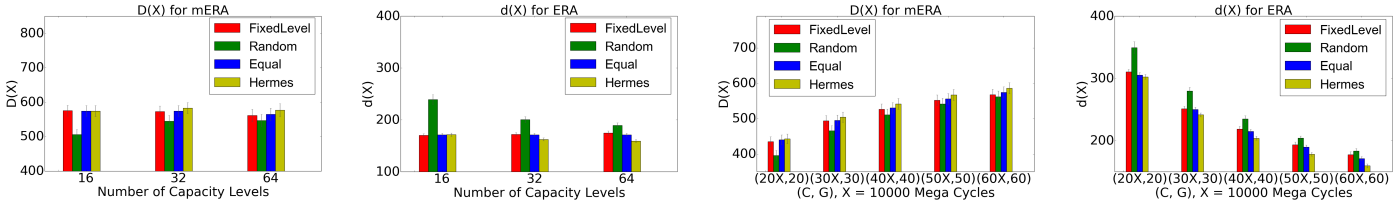
We introduce four algorithms for comparison. OPT: we simply use brute force to enumerate all possible allocations. Equal: $x_i = \frac{C}{n}$ for each $i \in [1, n]$. Random: the budget is randomly partitioned into n parts, which are assigned to n clouds, respectively. FixedLevel: we first compute the optimal x_i for each h_i irrespective of the other clouds, and then use Alg. 2 to find a selection.

Due to the high time complexity of OPT, we compared Hermes with OPT on a "small" setting: the number of edge servers, n, was 6, and the number of jobs, m, was 50. When transforming mERA into dmERA, we introduce the hyper-parameter G. We want to know how G affects the performance of Hermes with respect to mERA and ERA. Figs. 4(a) and 4(b) show the results under varying G. We made two main observations.



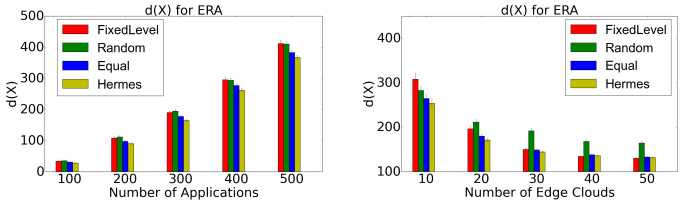
(a) Impact of G on mERA (b) Impact of G on ERA (c) Impact of C on mERA (d) Impact of C on ERA

Fig. 4: Simulation results on small instances (the default setting is $n = 6$ and $m = 50$)



(a) Impact of G on mERA (b) Impact of G on ERA (c) Impact of C on mERA (d) Impact of C on ERA

Fig. 5: Simulation results on large instances (the default setting is $n = 20$ and $m = 300$)



(a) Impact of the number of jobs when $n = 20$ (b) Impact of the number of edge servers when $m = 300$)

Fig. 6: ERA under varying m and n ($C = 600,000$, $G = 50$)

First, Hermes outperformed FixedLevel, Random, and Equal; the gap between Hermes and OPT was within 1% throughout these simulations. Second, when G increased, Hermes had more opportunities to improve the final allocation (i.e., $D(\mathbf{X})$ in mERA increased and $d(\mathbf{X})$ in ERA decreased when G increased).

We are also interested in evaluating the effect of the budget threshold. Figs. 4(c) and 4(d) depict the results. As we expected, the performance of all algorithms gets better when C increases, and the gap between Hermes and OPT is extremely small. Their close performance may help an ESP to make decisions when it wants to offer edge service but does not know how to choose a proper budget threshold.

We also ran Hermes and other algorithms except OPT on a "large" setting: the number of edge servers, n, was 20, and the number of jobs, m, was 300. Figs. 5 and 6 show the results. There is an interesting observation in Figs. 5(a) and 5(b). When the hyper-parameter G was less than 32, we found that, Equal outperformed Hermes. This is due to the fact that, when G was too small (e.g., G is less than $n = 20$ in these two figures), Hermes did not have much opportunity to improve the resource allocation, while Equal just split the total budget equally among n clouds and thus it was not affected by G. This

TABLE 2: The $\frac{\text{Hermes}}{\text{OPT}}$ ratio for ERA

m \ n	4	5	6	7	8
30	1.0063	1.0119	1.0203	1.0313	1.0443
40	1.0055	1.0104	1.0201	1.0321	1.0478
50	1.0061	1.0162	1.0226	1.0178	1.0383

observation suggests that we should choose for G a value at least larger than n.

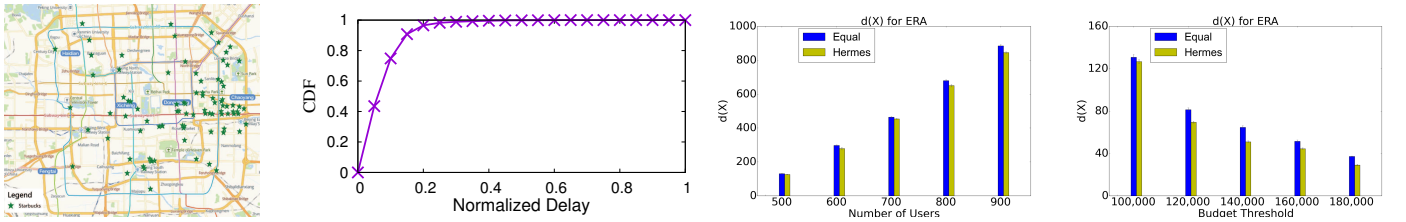
Figs. 5(c) and 5(d) show the effect of the budget threshold C in the large setting. We note that, throughout these simulations, Hermes always outperformed the other three baselines, and Hermes reduced the total makespan by 7% on average and 10% at most compared with the second-best one.

Fig. 6 demonstrates the impact of the number of jobs, m, and the number of edge servers, n. When the number of jobs increases, the total makespan increases; however, more jobs benefit from nearby edge servers and the individual makespan is shortened. When the number of edge servers increases, Hermes has more opportunities to minimize the total makespan.

To generalize our evaluation on approximation ratio, Table 2 shows the approximation ratio of Hermes when both of the number of edge servers and the number of jobs are varying. We found that the gap between Hermes and OPT was 2.21% on average, and 4.78% at most, which is far better than our theoretical bounds.

5.2 Trace-driven Evaluation

We consider a metropolitan area that contains edge servers and users in this paper. For locations of edge servers, we use the locations of Starbucks within the 4th ring road of Beijing, as shown in Fig. 7(a). Similar to a previous study [17], we use Starbucks' locations as the locations of edge servers, because the distribution of them in a city usually achieves a decent coverage of



(a) Locations of 92 Starbucks in Beijing, China.

(b) $y \times 100\%$ connections are associated with $x \times \%$ smallest delays.

(c) Impact of the number of jobs

(d) Impact of the budget threshold

Fig. 7: Trace-driven results (the default setting is $C = 100,000$ and $m = 500$)

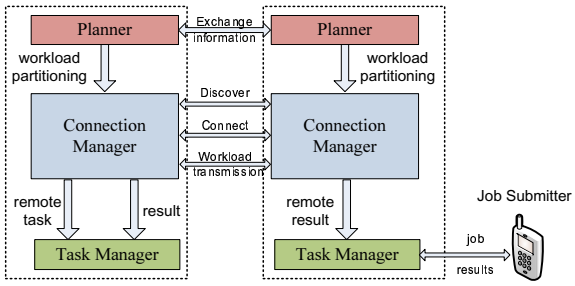


Fig. 8: Testbed for Hermes on 8 phones and 2 PCs

users, making them very suitable for placing edges. We calculate the minimum bounding rectangle of these 92 Starbucks with two sides parallel to a meridian. Then, we extend this rectangle by adding 5km to each side, so as to form the area of interest, within which we randomly generate user locations.

We assume there is a connection between a user and a server if the Euclidean distance between them is not larger than a pre-defined threshold, e.g., 5km; then, the propagation delays between them are synthesized following realistic distributions disclosed in [45], making sure the average delay is 20ms. The CDF of the synthetically generated delays is shown in Fig. 7(b), in which $y \times 100\%$ connections are associated with $x \times \%$ smallest delays. For example, 80% connections are associated with nearly 14% smallest delays in Fig. 7(b).

Similar to the simulations in the last subsection, we uniformly generated each b_j between 100 to 1000 Mega cycles per second; the amount of workloads of a job was uniformly generated between 500 to 1500 Mega cycles. By default, the number of jobs was 500; the budget threshold was 100,000 Mega cycles per second; and the hyper-parameter G was 20.

Figs. 7(c) and 7(d) shows the impact of the number of jobs and the budget threshold, in which the default setting is $m = 500$ and $C = 100,000$. We have similar observations as in simulations, although we consider heterogeneous propagation delays here. When the number of jobs increases, the amount of available resources a job can use decreases, making the average makespan increases; when the budget threshold increases, the ESP can rent more resources from EIPs, decreasing the average makespan of all jobs.

TABLE 3: The average makespan of the submitted jobs on our testbed

Servers \ Algs	Equal	FixedLevel	Hermes	OPT
2 phones	221.9ms	220.7ms	217.3ms	201.1ms
3 phones	195.8ms	193.2ms	182.9ms	162.5ms
4 phones	181.2ms	174.2ms	161.0ms	143.2ms
4 phones + 1 PC	163.5ms	159.9ms	132.1ms	110.7ms
4 phones + 2 PCs	146.2ms	136.4ms	112.2ms	92.1ms

5.3 Testbed-based Evaluation

We implemented Hermes on eight Android phones (Hisilicon Kirin 810 with 6GB bytes of memory) and two computers (2.3 GHz Intel Core i5 with 8G bytes of memory). Three phones were used as end users, one phone as the controller, and the rest of them were considered as edge servers. Hermes contained three main components, as shown in Fig. 8. The connection manager allowed neighbor discovery, pairwise connection, and data exchange. The task manager was responsible for managing tasks states, running data analysis tasks, and merging outputs/results. The planner communicated with other Android phones for necessary information (e.g., available computation resources) exchange. The controller was responsible for running the Hermes algorithm to split workloads of a job among phones.

As we mentioned in Section 3.1, there are many types of divisible workloads such as pattern search, file compression, joining operation in relational databases, graph coloring, and genetic search. We evaluated Hermes using the Karger's algorithm for the min-cut problem as the divisible job. To guarantee Karger's algorithm returns the min-cut with high probability, it is proved that Karger's algorithm should be executed no less than $\frac{|V|(|V|-1)}{2}$ times, where $|V|$ is the number of vertices in a graph. In our experiment, we randomly generated a graph with $|V| = 200$, thus, Karger's algorithm should be run at least 9,900 times on the graph. We see represent the size of the job as 9,900, and the minimal indivisible block is one run of the Karger's algorithm.

Table 3 shows the comparison results on our testbed in terms of average makespan of the submitted jobs. The results are averaged over 10 independent runs. Throughout the experiments, we found that the gap between Hermes and OPT was 21.82% at most, which is far better than our theoretical bounds.

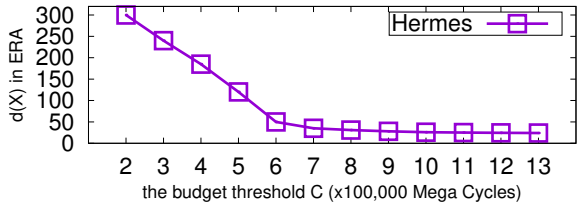


Fig. 9: The makespan-budget trade-off in Hermes

Fig. 9 illustrates the makespan-budget relationship in Hermes. When an ESP increases its budget threshold, the total makespan decreases; however, the returns are diminishing. This trade-off can be used by an ESP to choose the right balance between makespan and budget.

6 Conclusion and Future Work

In this paper, we study the problem of resource allocation for edge service entities under a budget threshold, identify its NP-completeness, and design an approximation solution—Hermes—through two-step transformation and theoretical analysis. The evaluations results confirm our theoretical findings and claims.

Hermes has several limitations, which are also the directions of our future work. Firstly, Hermes assumes the wireless connections between end users and edge servers are fixed, e.g., u_2 connects with h_1 and h_2 at all times. In reality, the connection between a user and an edge server may change over time, due to the physical motion of the user. A simple way to adapt to this changing situation is to re-run Hermes whenever a connection disappears or a new connection emerges.

Secondly, we do not consider hierarchical edge servers in this paper. With hierarchical edge servers, an edge server can further delegate its workloads to upper-layer edge servers (which are usually more powerful), while in our current design, an edge server cannot delegate its workload to others.

Lastly, the optimization goal (i.e., minimizing the average makespan of a set of jobs) may not represent the true need of individual users. A user may be more interested in reducing energy consumption of its mobile device. In the future, we may need to support more metrics from the user perspective.

Acknowledgments

This work was supported in part by NSFC (61872175), Natural Science Foundation of Jiangsu Province (BK20181252), the Fundamental Research Funds for the Central Universities (14380060), and Collaborative Innovation Center of Novel Software Technology and Industrialization. Sheng Zhang is the corresponding author.

References

[1] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *Proc. of ACM MobiSys 2013*, pp. 153–166.

[2] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. ACM MobiSys 2014*, pp. 68–81.

[3] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proc. Mobidata 2015*, pp. 37–42.

[4] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proc. ACM Workshop MCS 2010*, pp. 6–11.

[5] 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.

[6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[7] C. Wang, S. Zhang, H. Zhang, Z. Qian, and S. Lu, "Edge cloud capacity allocation for low delay computing on mobile devices," in *Proc. IEEE ISPA 2017*, pp. 1–8.

[8] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. IEEE INFOCOM 2016*, pp. 1–9.

[9] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253–2266, 2015.

[10] 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. ACM MobiSys 2010*, pp. 49–62.

[11] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proc. ACM EuroSys 2011*, pp. 301–314.

[12] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "COMET: code offload by migrating execution transparently," in *Proc. USENIX ODSI 2012*, pp. 93–106.

[13] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM 2012*, pp. 945–953.

[14] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE INFOCOM 2018*, pp. 1–9.

[15] Y. Bao, Y. Peng, C. Wu, and Z. Li, "Online job scheduling in distributed machine learning clusters," in *Proc. IEEE INFOCOM 2018*, pp. 1–9.

[16] R. Yu, G. Xue, and X. Zhang, "Application provisioning in fog computing-enabled internet-of-things: A network perspective," in *Proc. IEEE INFOCOM 2018*, pp. 1–9.

[17] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. IEEE INFOCOM 2018*, pp. 1–9.

[18] "Open Edge Computing," http://openedgecomputing.org/.

[19] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *arXiv preprint arXiv:1605.08518*, 2016.

[20] H. Tan, Z. Han, X.-Y. Li, and F. C. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. IEEE INFOCOM 2017*, 2017, pp. 1–9.

[21] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint," in *Proc. IEEE INFOCOM 2018*, pp. 1–9.

[22] N. Chen, S. Zhang, Z. Qian, J. Wu, and S. Lu, "When learning joins edge: Real-time proportional computation offloading via deep reinforcement learning," in *Proc. IEEE ICPADS 2019*, pp. 1–8.

[23] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proc. of IEEE INFOCOM 2016*, pp. 1–9.

[24] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Prof. of IEEE INFOCOM 2018*, pp. 1–9.

[25] L. Zhang and X. Tang, "Client assignment for improving interactivity in distributed interactive applications," in *Proc. IEEE INFOCOM 2011*, pp. 3227–3235.

[26] Y. Liang, J. Ge, S. Zhang, J. Wu, Z. Tang, and B. Luo, "A utility-based optimization framework for edge service entity caching," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–12, 2019.

[27] Y. Liang, J. Ge, S. Zhang, J. Wu, L. Pan, T. Zhang, and B. Luo, "Interaction-oriented service entity placement in edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–12, 2019.

[28] V. Vazirani, *Approximation algorithms*. Springer, 2004.
[29] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE INFOCOM 2018*, pp. 1–9.
[30] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *Proc. USENIX NSDI 2017*, pp. 377–392.
[31] J. Jiang, G. Ananthanarayanan, P. Bodk, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. ACM SIGCOMM 2018*, pp. 1–14.
[32] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. IEEE INFOCOM 2020*, pp. 1–10.
[33] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proc. ACM workshop MCS 2012*, pp. 29–36.
[34] S. Zhang, J. Wu, and S. Lu, "Distributed workload dissemination for makespan minimization in disruption tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1661 – 1673, 2016.
[35] Y. Li and W. Wang, "Can mobile cloudlets support mobile applications?" in *Proc. IEEE INFOCOM 2014*, pp. 1060–1068.
[36] B. P. Rimal, D. P. Van, and M. Maier, "Cloudlet enhanced fiber-wireless access networks for mobile-edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3601–3618, 2017.
[37] V. Bharadwaj, D. Ghose, and T. G. Robertazzi, "Divisible load theory: a new paradigm for load scheduling in distributed systems," *Cluster Computing*, vol. 6, no. 1, pp. 7–17, 2003.
[38] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. ACM SIGKDD 2007*, pp. 420–429.
[39] S. Fujishige, *Submodular functions and optimization*. Elsevier, 2005, vol. 58.
[40] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.
[41] L. Li, R. Geda, A. B. Hayes, Y. Chen, P. Chaudhari, E. Z. Zhang, and M. Szegedy, "A simple yet effective balanced edge partition model for parallel computing," in *Proc. ACM SIGMETRICS 2017*, pp. 1–21.
[42] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, March 2018.
[43] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE INFOCOM 2016*, pp. 1–9.
[44] D. Yang, X. Fang, and G. Xue, "ESPN: Efficient server placement in probabilistic networks with budget constraint," in *Proc. IEEE INFOCOM 2011*, pp. 1269–1277.
[45] T. Høiland-Jørgensen, B. Ahlgren, P. Hurtig, and A. Brunstrom, "Measuring latency variation in the internet," in *Proc. ACM CoNEXT 2016*, pp. 473–480.

Sheng Zhang (M'14) is an associate professor in the Department of Computer Science and Technology, Nanjing University. He is also a member of the State Key Lab. for Novel Software Technology. He received the BS and PhD degrees from Nanjing University in 2008 and 2014, respectively. His research interests include cloud computing and edge computing. To date, he has published more than 70 papers, including those appeared in *TMC*, *TON*, *TPDS*, *TC*, *MobiHoc*, *ICDCS*, *INFOCOM*, *SECON*, *IWQoS*, and *ICPP*. He received the Best Paper Runner-Up Award from IEEE MASS 2012. He is the recipient of the 2015 ACM China Doctoral Dissertation Nomination Award. He is a member of the IEEE and a senior member of the CCF.

Yu Liang is a PhD candiate in Nanjing University. She received her MS degree from Nanjing University in 2011. She was a senior software engineer in Trend Micro China Development Center between 2011 and 2017. Her research interests include resource allocation in cloud and edge computing. Her publications include those appeared in TPDS, TMC, COMNET, COMCOM, ICDCS, and Globecom.

Jidong Ge is an associate Professor at Software Institute, Nanjing University. He received his PhD degree in Computer Science from Nanjing University in 2007. His current research interests include cloud computing, distributed computing, workflow scheduling, workflow modeling, process mining. His research results have been published in more than 90 papers in international journals and conference proceedings including IEEE TPDS, IEEE TSC, JPDC, COMNET, JASE, JNCA, FGCS, JSS, Inf. Sci., ESA, ICSE, ASE, IWQoS, GlobeCom, APSEC, ICSSP, HPCC, SEKE etc.

Mingjun Xiao is an associate professor in the School of Computer Science and Technology at the University of Science and Technology of China (USTC). He received his Ph.D. from USTC in 2004. In 2012, he was a visiting scholar at Temple University, under the supervision of Dr. Jie Wu. He is a TPC member of many conferences, including IEEE INFOCOM 2018, IEEE ICDCS 2015, ACM Mobihoc 2014, etc, and has served as a reviewer for many journal papers. His main research interests include mobile crowdsensing, mobile social networks, and vehicular ad hoc networks. He has published over 50 papers in refereed journals and conferences, including TON, TMC, TPDS, TC, INFOCOM, etc.

Jie Wu (F'09) is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Mobile Computing, IEEE Transactions on Service Computing, Journal of Parallel and Distributed Computing, and Journal of Computer Science and Technology. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.