Efficient Task Scheduling With Stochastic Delay Cost in Mobile Edge Computing

Wenyu Zhang, Zhenjiang Zhang, Sherali Zeadally, and Han-Chieh Chao

***Abstract*— We propose a new efficient and effective task scheduling approach with stochastic time cost for computation offloading in mobile edge computing. We developed an optimization model that minimizes the maximum tolerable delay (MTD) by considering both the average delay and delay jitter. We also proposed an efficient conservative heterogeneous earliest-finish-time algorithm to solve the MTD-minimization problem. Numerical results obtained with our proposed approach demonstrate its effectiveness over previously proposed techniques.**

***Index Terms*— Call graph, delay jitter, mobile edge computing, scheduling, stochastic delay.**

I. Introduction

In recent years, mobile edge computing (MEC) has attracted a lot of attention because of its benefits such as low-latency computation and caching services to mobile user terminals. Computation offloading is one core function in MEC that enables user terminals to offload their computation intensive tasks to MEC servers while at the same time reduces the energy cost of energy-constrained terminals [1].

Before offloading the tasks to MEC servers, a task scheduling or code partitioning operation needs to be conducted to minimize offloading costs such as delay time and energy consumption. In software programming, an application is usually composed of several functions organized by a directed call graph (DCG). A member function corresponds to a task that needs to be executed, and the DCG shows the execution dependencies of the tasks. The time cost and energy cost of the application are influenced by the execution orders and the execution locations of the tasks which need to be properly assigned before the tasks are executed.

For delay-optimal task scheduling in MEC, current works [1], [2] mainly consider the static task scheduling problem for data processing. Liu *et al.* [3]modeled the task scheduling problem as a Markov chain decision process, and they proposed a stochastic scheduling model to find the optimal decision probabilities. Mao *et al.* [4] considered energy harvesting in the task scheduling process, and they proposed a Lyapunov-based optimization approach to achieve optimal task scheduling with minimum time cost. Khalili and Simeone [5] and Kiani and Ansari [6], the authors considered the impact of DCG on task scheduling, and they proposed models that jointly optimize the transmission rates and offloading decisions.

Manuscript received October 11, 2018; revised October 25, 2018; accepted October 29, 2018. Date of publication November 5, 2018; date of current version January 8, 2019. This work is supported by National Natural Science Foundation of China under grant 61772064, and Academic Discipline, Post-Graduate Education Project of the Beijing Municipal Commission of Education, and Fundamental Research Funds for the Central Universities under grant 2017YJS026. The associate editor coordinating the review of this paper and approving it for publication was J. Ben Othman. *(Corresponding author: Zhenjiang Zhang.)*

W. Zhang and Z. Zhang are with the Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: zhangzhenjiang@bjtu.edu.cn).

S. Zeadally is with the College of Communication and Information, University of Kentucky, Lexington, KY 40506 USA.

H.-C. Chao is with the Department of Electrical Engineering, National Dong Hwa University, Hualien 974, Taiwan.

Digital Object Identifier 10.1109/LCOMM.2018.2879317

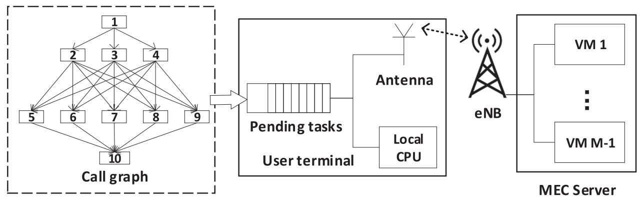

Fig. 1. System model.

In the above related works, although DCG has been previously considered in modeling the time cost, the task execution orders are not considered. Most importantly, in addition to the average delay, the delay-jitter is also an important factor that affects the quality of service (QoS) of MEC, and needs to be carefully considered. Moreover, most of the related works considered static delay cost on computation and transmission which are not consistent with the stochastic nature of delay time in MEC systems.

We summarize the contributions of this work as follows: (1) We define two performance metrics called delay risk probability (DRP) and maximum tolerable delay (MTD), which consider both the average delay time and the stochastic delay jitter. (2) We developed an optimization model that aims to minimize MTD in order to solve the task scheduling problem with stochastic delay cost, and we propose an improved conservative heterogeneous earliest-finish-time (CHEFT) algorithm to solve the MTD-minimization problem wherein the probability density estimation is achieved by using the kernel density estimation (KDE) method. The original HEFT algorithm is an acknowledged well-known, efficient and effective algorithm for solving static task scheduling problems [7]. (3) We also proposed a Gaussian approximated CHEFT algorithm to reduce the complexity of probability density estimation process. Our simulation results demonstrate that the proposed CHEFT method outperforms static HEFT algorithm in MEC systems with stochastic delay costs.

II. System Model

As shown in Fig. 1, we consider a task scheduling problem in a MEC system with multiple computing locations denoted by the set $\mathcal{L} = \{l_1, \ldots, l_M\}$. By default, we set l_1 as the computation location of user terminal, and the other $M-1$ parallel computing virtual machines (VMs) are located in the MEC server. For an application to be executed, its call graph is represented as a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{Z}, \mathcal{B}\}$, in which $\mathcal{V} = \{v_1, \ldots, v_N\}$ denotes the set of tasks, $\mathcal{Z} = \{z_1, \ldots, z_N\}$ is the

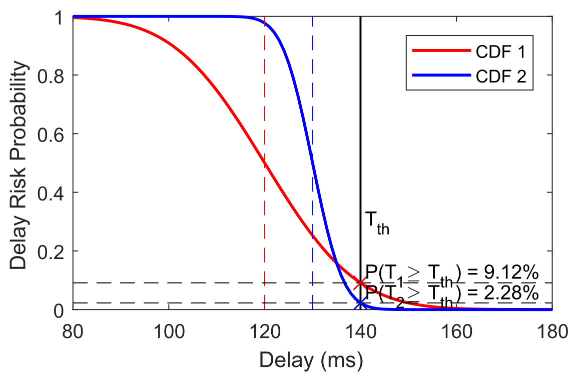

Fig. 2. An example of delay risk probability.

set of corresponding computation demands, and the set $\mathcal{B} = \{b_{i,j}\}_{N\times N}$ contains the amount of transmitted data among the tasks. If task v_j is an immediate successor of task v_i, we have $b_{i,j} > 0$, otherwise $b_{i,j} = 0$.

In addition to the average delay, the QoS of MEC is also significantly influenced by the delay jitter of the task offloading process. If the fluctuation of delay jitter is too large, the service will be unacceptable even if the average delay is low. As such, we define delay risk probability (DRP) as the probability that the total delay time is larger than a predefined tolerable threshold T_{th}, as given by

$$P(T \geq T_{th}) = \int_{T_{th}}^{\infty} p(T)dT = R(p(T), T_{th}), \tag{1}$$

where $R(p(x), x_0)$ denotes the risk probability function at point x_0 with respect to the probability density function (PDF) $p(x)$. In contrast, if a target risk probability function P_{th} is given, we have the maximum tolerable delay (MTD) as

$$T^{mt} = R^{-1}(P_{th}, p(T)). \tag{2}$$

To provide a better understanding of DRP and MTD, Fig. 2 illustrates an example which plots the complementary cumulative distribution functions (CCDFs) of two Gaussian-distributed delays T_1 and T_2. The two average delays are $\bar{T}_1 = 120\ ms$ and $\bar{T}_2 = 130ms$, but the corresponding standard deviations are $\sigma_1 = 15ms$ and $\sigma_1 = 5ms$, which means that the delay jitter of T_1 is larger than T_2. We note when the MTD $T_{th} = 140ms$, the DRPs of T_1 and T_2 are 9.12% and 2.28% respectively. In the same way, given a DRP that is larger than 0.1, the corresponding MTD of T_2 is smaller than T_1. In fact, if we use DRP or MTD as the performance metric, T_2 is better than T_1 in this scenario.

We denote the execution locations and the execution orders of the tasks as $\mathbf{a} = [a_i]_{1\times N}$ and $\mathbf{d} = [d_i]_{1\times N}$ respectively. Our goal is to minimize the total MTD with a given target risk probability by assigning appropriate scheduling decisions and execution orders of the tasks, as given by:

$$\min_{\mathbf{a},\mathbf{d}}\ R^{-1}(P_{th}, p(T)) \tag{3}$$

subject to:

$$a_i \in \{1, \cdots N\}, \text{ and } a_i \neq a_j\ (\forall i \neq j) \tag{4}$$

$$a_i < a_j, \text{ if } v_i \text{ is a predecessor of } v_j \tag{5}$$

$$d_i \in \{1, \cdots M\} \tag{6}$$

Constraint (4) ensures that the tasks are executed with different orders, constraint (5) guarantees that a task must be executed after its predecessor tasks, constraint (6) means that a task can be executed at any of the computation locations. It is worth noting that we assume that the energy cost on data transmission imposed by computation offloading is acceptable, therefore we do not consider it in the above model.

III. Efficient Task Scheduling

Since the explicit objective function is not available, the optimization model needs to be solved by using a heuristic algorithm. However, implementing a heuristic algorithm with high complexity incurs higher delays and energy costs, which may not be acceptable in practice. In this work, we propose an efficient conservative HEFT (CHEFT) algorithm to find a solution for the MTD-minimization problem. We describe the algorithm in the following subsections.

A. PDF Estimation

For the same task, its data transmission time and execution time are all stochastic parameters. By collecting the historical transmission and computation time data, their probability density distributions (PDFs) can be obtained by using a PDF estimation method. Suppose that we have K historical delay time samples $\mathbf{t} = [t_1, \ldots, t_K]$. We use the kernel density estimation (KDE) method [8] to obtain the target PDF which is given by

$$p(t) = \frac{1}{\lambda K}\sum_{i=1}^{K}\mathcal{K}\left(\frac{t-t_i}{\lambda}\right), \tag{7}$$

where $\mathcal{K}(\cdot)$ is the kernel function, and λ is the bandwidth parameter. Usually, the Gaussian kernel function [8], i.e. $\mathcal{K}(\frac{t-t_i}{\lambda}) = exp(-\frac{(t-t_i)^2}{\lambda})$, is used as the base kernel. According to Silverman's rule of thumb, the parameter λ can be empirically calculated from $\lambda = \sigma\left(\frac{4}{3K}\right)^{0.2} \approx 1.06\sigma K^{-0.2}$, where σ is the estimated standard deviation of $\mathbf{t}$. With the estimated PDF, the corresponding MTD can be obtained by using equation (2).

B. Task Execution Orders and Locations

We denote t_i^{cp} (s/GHz) as the computation time cost of 1-GHz computation demand that is executed in location l_i. In the same way, we denote $t_{i,j}^{tr}$ as the transmission time cost of 1-kbytes of data transmitted from location l_i to location l_j. Similar to the HEFT algorithm, we use the upward rank values to assign the execution orders of the tasks. We define the conservative upward rank (CUR) of task v_i as:

$$\mathit{CUR}(v_i) = z_i\bar{t}^{mtcp} + \max_{v_j\in su\{v_i\}}\{b_{i,j}\bar{t}^{mttr} + \mathit{CUR}(v_j)\}, \tag{8}$$

with $\bar{t}^{mtcp} = \frac{1}{M}\sum_{i=1}^{M}R^{-1}(P_{th}, p(t_i^{cp}))$, and $\bar{t}^{mttr} = \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M}R^{-1}(P_{th}, p(t_{i,j}^{tr}))$, where $su\{v_i\}$ is the set that contains the immediate successors of v_i. For the exit task without successors, its CUR is $rank_{cu}(v_{eixt}) = z_{exit}\bar{t}^{mtcp}$. By traversing the DCG upward from the exit task, the CURs of all the tasks can be obtained, and then the task priority phase can be executed by sorting the CURs in decreasing order. The priorities obtained will be considered as the execution orders of the tasks.

Having obtained the task priorities, the algorithm then starts the execution locations¡¯ selection phase. Similar to HEFT, the task is executed on the processor with the earliest available time, while preserving the call graph constraint. More specifically, the earliest finish time (EFT) and earliest start time (EST) time of task v_i are defined as:

$$\mathit{EST}(v_i, l_j) = \max\left\{\mathit{ava}(l_j), \max_{v_m \in \mathit{pre}\{v_i\}} \{arr(v_m, l_j)\}\right\}, \quad (9)$$

$$\mathit{EFT}(v_i, l_j) = z_i t_j^{cp} + \mathit{EST}(v_i, l_j), \quad (10)$$

where $\mathit{ava}(l_j)$ denotes the earliest time that location l_j is ready for computation, $\mathit{pre}\{v_i\}$ is the set that contains the immediate predecessors of v_i, $arr(v_m, l_j)$ means the arrival time of task v_m which can be calculated by $arr(v_m, l_j) = AFT(v_m) + b_{m,j} t_{m,j}^{tr}$, where $AFT(v_m)$ is short for the actual finish time (AFT) of task v_m, and it is equal to the EFT when the actual computation location of v_m is selected. Equation (9) means that, when the predecessor tasks are all completed, $\mathit{EST}(v_i, l_j)$ is equal to the earliest available time of location l_j, otherwise $\mathit{EST}(v_i, l_j)$ is equal to the maximum timespan of its predecessor tasks. Equation (10) means that $\mathit{EFT}(v_i, l_j)$ can be directly obtained by adding the task computation time when $\mathit{EST}(v_i, l_j)$ is known.

The above formulations only consider the average delay of the tasks, but ignore their delay jitters. We therefore define the maximum tolerable EFT (MTEFT), as given by:

$$\mathit{MTEFT}(v_i, l_j) = R^{-1}(P_{th}, p(\mathit{EFT}(v_i, l_j))). \quad (11)$$

Given a new pending task, it will be assigned to a computation location with the minimum MTEFT, as given by:

$$a_i = \arg\min_{l_j \in \mathcal{L}} \{\mathit{MTEFT}(v_i, l_j)\}. \quad (12)$$

Based on the above modeling, we propose a conservative HEFT algorithm for task scheduling with stochastic time cost in MEC, as shown in Algorithm 1.

Algorithm 1 The CHEFT Algorithm

```
1: Initialization: The target DRP $P_{th}$, the DCG $\mathcal{G}$.
2: Estimate related PDFs by using Eq. (7). Calculate $CUR$s
   by using Eq. (8).
3: Sort $CUR$s in decreasing order and obtain task execution
   priority vector $\mathbf{d} = [d_i]_{1\times N}$.
4: for $i = 1 : 1 : N$
5:   for $j = 1 : 1 : M$
6:     Estimate related PDFs by using Eq. (7).
7:     Calculate $MTEFT(v_{d_i}, l_j)$ by using Eqs. (9), (10) and
   (11).
8:   end for
9:   Assign execution location $a_{d_i}$ by using Eq. (12).
10: end for
```

C. Gaussian Approximation

The estimation of high-precision PDFs and the calculation of inverse risk probability function suffer from high computation complexity. As suggested in [9], we assume that the delay follows a Gaussian distribution to reduce the calculation complexity. In this way, equation (2) becomes

$$t^{mt} = \sigma^2 \; Q^{-1}(P_{th}) + \mathbb{E}(t), \quad (13)$$

where σ^2 is the variance of t, and $Q(\cdot)$ is the CCDF function of standard Gaussian distribution, i.e. $Q(x) = \frac{1}{2\pi}\int_x^\infty \exp(\frac{-x^2}{2})dx$. In this way, $\bar{t}^{mtcp}$ and $\bar{t}^{mttr}$ in equation (8) can be directly calculated by:

$$\bar{t}^{mtcp}_{gau} = \frac{1}{M}\sum_{i=1}^{M}((\sigma_k^{cp})^2 \; Q^{-1}(P_{th}) + \mathbb{E}(t_i^{cp})), \quad (14)$$

$$\bar{t}^{mttr}_{gau} = \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M}((\sigma_{i,j}^{tr})^2 \; Q^{-1}(P_{th}) + \mathbb{E}(t_{i,j}^{tr})), \quad (15)$$

where $(\sigma_k^{cp})^2$ and $(\sigma_{i,j}^{tr})^2$ are the variances of t_i^{cp} and $t_{i,j}^{tr}$ respectively. $\mathbb{E}(x)$ denotes the expectation of random variable x. In the same way, equation (11) is

$$\begin{aligned}\mathit{MTEFT}(v_i, l_j) &= (\sigma_{i,j}^{EFT})^2 \; Q^{-1}(P_{th})) + \mathbb{E}(\mathit{EFT}(v_i, l_j)) \\ &= \mathit{EFT}(v_i, l_j)|_{t_{ij}^{tr} = \bar{t}^{mttr}_{gau}, t_j^{cp} = \bar{t}^{mtcp}_{gau}}, \end{aligned} \quad (16)$$

where $(\sigma_{i,j}^{EST})^2$ is the variance of $\mathit{EFT}(v_i, l_j)$. The second line of equation (16) means that there is no need to estimate the variance for each $\mathit{MTEFT}(v_i, l_j)$, and we can directly calculate $\mathit{EFT}(v_i, l_j)$ by replacing t^{cp} and t^{tr} with their corresponding t^{mtcp} and t^{mttr} respectively.

Algorithm 2 The Gaussian Approximated CHEFT Algorithm

```
1: Initialization: The target DRP $P_{th}$, the DCG $\mathcal{G}$.
2: Calculate $\mathbb{E}(\mathbf{t}^{tr}) = [\mathbb{E}(t_i^{tr})]_{1\times M}$, $\boldsymbol{\sigma}^{tr} = [\sigma_i^{tr}]_{1\times M}$, and
   $\mathbb{E}(\mathbf{t}^{cp}) = [\mathbb{E}(t_i^{cp})]_{1\times M}$, $\boldsymbol{\sigma}^{cp} = [\sigma_i^{tr}]_{1\times M}$. Calculate the
   corresponding MTDs using Eq. (13).
3: Calculate $CUR$s by using Eq. (8) with the MTDs.
4: Sort $CUR$s in decreasing order and obtain task execution
   priority vector $\mathbf{d} = [d_i]_{1\times N}$.
5: for $i = 1 : 1 : N$
6:   for $j = 1 : 1 : M$
7:     Calculate $MTEFT(v_{d_i}, l_j)$ by using Eq. (16).
8:   end for
9:     Assign execution location $a_{d_i}$ by using Eq. (12).
10: end for
```

Algorithm 2 presents the Gaussian approximated CHEFT algorithm. The complexity of KDE-CHEFT is about $\mathcal{O}(MNK^2)$, while the Gaussian-CHEFT is about $\mathcal{O}(MNK)$, which is much smaller. It is worth pointing out that the complexity of KDE is $\mathcal{O}(K^2)$, thus the number of data samples K should not be too large if KDE is used. By using Gaussian-CHEFT, we can reduce the scheduling complexity if the performance loss is acceptable.

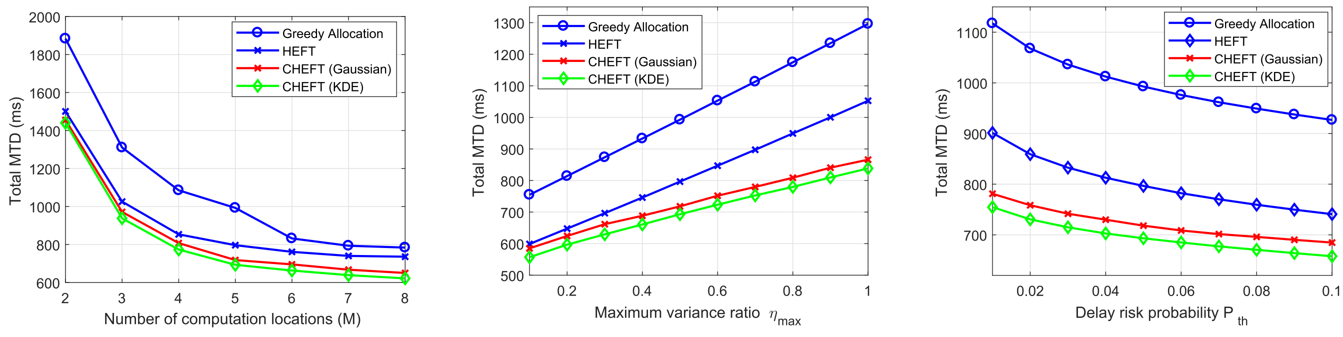

Fig. 3. Performance comparison of EFT, HEFT and the proposed CHEFT, in which (a), (b) and (c) plot the total MTDs with different number of computation locations M, maximum variance ratio η_{max}, and risk probability P_{th}, respectively.

IV. Simulation Results

In this section we evaluate the performance of our proposed method by using the example application shown in Fig. 1 which includes 10 different tasks. We conducted the simulation tests in an environment consisting of Intel i7 2.4GHz CPU, 8GB RAM, Matlab 2017a, and Win 10 operating system.

In addition to the conventional HEFT algorithm, the results of a greedy task allocation method are also provided for performance comparison. Similar to HEFT, with greedy allocation, a task is assigned with a computation location with minimum EFT, but the execution orders are not considered. The data amount of a task is randomly generated from 100 to 800 kbytes, and their computation demands are randomly distributed from 0.1 to 2GHz. The average computation speeds of the user terminal and VMs are 1s/GHz and 0.25s/GHz respectively. The average data transmission rate between two different computation locations is 100 kbytes/s. The computation and transmission times are randomly generated by using a PDF that is approximated to a Gaussian distribution $t \sim \mathcal{N}(\mu_t, \sigma_t^2)$, but a small portion of them follow a chi-square distribution $t \sim \chi^2(\mu_t)$. We define the maximum variance ratio $\eta_{max} = \sigma^2/\mu_t$ to control the variances of the delays, i.e. the variances of the related delays are randomly generated from 0 to $\eta_{max}\mu_t$. The default values of the risk probability P_{th}, maximum variance ratio $\eta_{max} = \sigma^2/\mu_t$ and number of computation locations M are set to 0.05, 0.5 and 5 respectively. To estimate the PDF, we randomly generate $K = 500$ samples for each delay cost, and we repeated the simulations 100 times to obtain the average performance comparison results.

The performance comparison results are shown in Fig. 3, in which subfigures (a), (b) and (c) plot the MTDs with different number of computation locations (M), maximum variance ratio η_{max}, and risk probability P_{th} respectively. From the above results, we note that the proposed CHEFT method outperforms the conventional HEFT and the greedy allocation methods in all of the above three scenarios. We observe that MTDs of greedy allocation is much higher than HEFT and CHEFT thereby demonstrating that considering the task execution order is necessary for the task scheduling problem with DCG. In particular, the results also demonstrate that a larger number of VMs is helpful in reducing delays of task offloading with DCGs. In addition, since a larger η_{max} means high delay jitter, we observe that the MTD reduction becomes much more evident when the delay jitter increases. The DRP also has an impact on the MTD: a larger DRP corresponds to a smaller reduction of MTD with the CHEFT algorithm. In all scenarios, the KDE-CHEFT is slightly better than the Gaussian-CHEFT, but the improvement is not significant. Considering the complexity of KDE, it is better to use Gaussian-CHEFT when the distributions of delays are approximated to Gaussian distributions.

V. Conclusion

We studied the MTD-minimal task scheduling problem in MEC, which considers both the total delay time and the delay-jitter. To efficiently and effectively solve the scheduling problem, we have proposed a CHEFT algorithm that is based on the well-known HEFT algorithm. We have also proposed a low-complexity Gaussian approximated CHEFT algorithm. The simulation results demonstrate the effectiveness of the proposed method. In our future work, we will consider the energy cost in the MTD-minimization model.

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.
[2] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative fog computing for dealing with big data in the Internet of vehicles: Architecture and hierarchical resource management," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 60–67, Dec. 2017.
[3] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2016, pp. 1451–1455.
[4] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.
[5] S. Khalili and O. Simeone, "Inter-layer per-mobile optimization of cloud mobile computing: A message-passing approach," *Trans. Emerg. Telecommun. Technol.*, vol. 27, no. 6, pp. 814–827, 2016.
[6] A. Kiani and N. Ansari, "Optimal code partitioning over time and hierarchical cloudlets," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 181–184, Jan. 2018.
[7] H. Zhao and R. Sakellariou, "An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm," in *Proc. Eur. Conf. Parallel Process.*, 2003, pp. 189–194.
[8] W. Zhang, Z. Zhang, H.-C. Chao, and F.-H. Tseng, "Kernel mixture model for probability density estimation in Bayesian classifiers," *Data Mining Knowl. Discovery*, vol. 32, no. 5, pp. 675–707, May 2018.
[9] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2867–2876, Nov. 2014.