Constrained Multi-objective Optimization for IoT-enabled Computation Offloading in Collaborative Edge and Cloud Computing

Guang Peng, Huaming Wu, *Member, IEEE,* Han Wu, *Student Member, IEEE,* and Katinka Wolter, *Associate Member, IEEE*

***Abstract*—Internet of things (IoT) applications are becoming more resource-hungry and latency-sensitive, which are severely constrained by limited resources of current mobile hardware. Mobile cloud computing (MCC) can provide abundant computation resources, while mobile edge computing (MEC) aims to reduce the transmission latency by offloading complex tasks from IoT devices to nearby edge servers. It is still challenging to satisfy the quality of service (QoS) with different constraints of IoT devices in a collaborative MCC and MEC environment. In this paper, we propose three constrained multi-objective evolutionary algorithms (CMOEAs) for solving IoT-enabled computation offloading problems in collaborative edge and cloud computing networks. First of all, a constrained multi-objective computation offloading model considering time and energy consumption is established in the mobile environment. Inspired by the push and pull search (PPS) framework, three constrained multi-objective evolutionary algorithms are developed by combing the advantages of population-based search algorithms with flexible constraint handling mechanisms. On one hand, three popular and challenging constrained benchmark suites are selected to test the performance of the proposed algorithms by comparing them to the other seven state-of-the-art CMOEAs. On the other hand, a multi-server multi-user multi-task computation offloading experimental scenario with a different number of IoT devices is used to evaluate the performance of three proposed algorithms and other compared algorithms as well as representative offloading schemes. The experimental results of the benchmark suites and computation offloading problems demonstrate the effectiveness and superiority of the proposed algorithms.**

***Index Terms*—Mobile edge computing, mobile cloud computing, computation offloading, Internet of Things (IoT), constrained multi-objective optimization.**

I. Introduction

With the explosive development of mobile networks and Internet of Things (IoT), more and more computation-intensive and latency-sensitive applications are emerging and deployed into different IoT devices [1], [2]. However, due to the inherent size constraints of IoT devices, limited computation capability and battery life cannot satisfy the quality of service (QoS) of these complex applications, such as augmented reality (AR), face recognition and online gaming [3]. Since the cloud servers have more powerful computation resources than mobile devices, the computation tasks can be offloaded to and processed at cloud servers, which can enhance the computation capacity and reduce energy consumption of these mobile devices [4]. The new computing paradigm that offloads tasks to the cloud through wireless networks is known as mobile cloud computing (MCC) [5].

Generally speaking, in MCC, cloud data centers are mostly a little far away from mobile devices, which need more propagation delay to the remote cloud. To address this problem, mobile edge computing (MEC) (or multi-access edge computing) [6] is a promising technique to overcome these challenges. In MEC, the edge servers are deployed at the edge of cellular networks, such as smart gateways, access points, and base stations [7]. The latency-sensitive tasks can be offloaded to edge servers with the aim to reduce the communication delay between mobile devices and edge servers. Hence, computation offloading is an attractive and challenging topic in MEC. A variety of architectures and offloading policies have been investigated. The literature [8] presented the challenges and methods of realizing low latency and high reliability of several mission-critical applications in MEC, such as virtual reality (VR), vehicle-to-everything (V2X), edge artificial intelligence (AI). Pham et al. [9] provided a holistic overview of MEC technology and its potential use cases and applications under the 5G mobile networks. Wang et al. [10] analyzed different architectural design alternatives based on cloud/edge/fog computing for connected vehicles. They also compared the characteristics in different edge computing paradigms, including MCC, Cloudlet, Fog computing and MEC.

Wu et al. [11] studied how to dynamically partition a given application and determine whether the computation task is executed locally or offloaded to edge/cloud servers. They proposed a min-cost offloading partitioning (MCOP) algorithm from the graph theory to reduce execution time and energy consumption. Dinh et al. [12] designed an offloading framework of a single mobile device and multiple edge nodes, and considered two cases for the mobile device's fixed and elastic CPU frequency. They proposed a linear relaxation based approach and a semidefinite relaxation (SDR) based approach for fixed and elastic CPU frequency cases, respectively. Bi et al. [13] put forward a joint optimization of service caching placement and computation offloading in mobile edge computing systems, a sequential task execution model is set up in a single-user with the assistance of a single-

G. Peng, H. Wu and K. Wolter are with Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany (e-mail: {guang.peng,han.wu,katinka.wolter}@fu-berlin.de). (Corresponding author: Guang Peng)

H. Wu is with Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

This work is supported by the National Natural Science Foundation of China (61801325, 62071327).

server. Wang et al. [14] proposed a cooperative task offloading model to minimize task duration with the constraint of energy consumption in three-tier mobile computing networks, and utilized alternating direction method of multipliers (ADMM) method to solve the problem. Du et al. [15] defined the cost as a weighted sum of latency and energy consumption of computation offloading problem in a mixed fog/cloud system, which considers the allocation of both computation resource and radio bandwidth, and the final objective minimizes the maximum cost among all users to guarantee the fairness for all users.

Some other work used deep learning methods to solve offloading problems [16]–[18]. Huang et al. [19] formulated the optimization problem of joint offloading decision and bandwidth allocation in MEC network, in which multiple wireless devices choose to offload their computation tasks to an edge server. Then they proposed a distributed deep learning-based offloading (DDLO) algorithm to generate near-optimal offloading decisions. Wu et al. [20] established the task offloading model with the aim to reduce latency and save energy in the collaboration of MCC and MEC, and proposed a distributed deep learning-driven task offloading (DDTO) algorithm to solve the offloading problems. Huang et al. [21] also investigated online computation offloading problems in the wireless powered mobile edge computing networks, and developed a deep reinforcement learning-based online offloading (DROO) framework to learn the binary offloading decisions from experience. Wang et al. [22] observed that many deep reinforcement learning (DRL) based methods have weak adaptability to new environments since they need full retraining to learn updated policies due to new environments. Hence, they proposed a meta reinforcement learning method to adapt fast to new environments with a relevant small number of gradient updates and samples.

On the other hand, multiple metaheuristic optimization algorithms have also received attention. Kuang et al. [23] established a system model in the MEC environment with multiple users, multiple end nodes, and structured tasks. Then they formalized an offloading decision problem as a cost-minimization problem and designed an improved genetic algorithm (GA) to solve that. Xu et al. [24] proposed a non-dominated sorting genetic algorithm III (NSGA-III) to address the multi-objective optimization problem of task offloading for cloudlet and cloud computing. Goudarzi et al. [25] investigated an application placement technique for concurrent IoT applications in edge and fog computing environments, and obtained a memetic algorithm (MA) algorithm based on the GA and one local search method to solve the offloading problems.

Computation offloading problems are often constrained optimization problems and NP-hard [14], [20], [25]. However, there are few studies that combine constrained multi-objective optimization with computation offloading in collaborative MCC and MEC. The motivation of this paper is to treat the computation offloading problem as a constrained multi-objective optimization problem (CMOP) and then we focus on the state-of-the-art constrained multi-objective evolutionary algorithms (CMOEAs) for solving that. A key issue in CMOEA is to deal with constraints. The penalty function approach is often used to balance objectives and constraints, which converts a CMOP into an unconstrained MOP by adding the overall constraint violation multiplied by a predefined penalty factor to each objective [26]. The constrained NSGA-II [27] adopted the constraint dominance principle to distinguish feasible and infeasible solutions. MOEA/D-IEpsilon [28] combined an improved epsilon constraint handling mechanism with a decomposition-based multi-objective evolutionary algorithm (MOEA/D) [29] to solve CMOPs. C-TAEA [30] maintained convergence-oriented archives and diversity-oriented archives simultaneously to retain the balance between convergence and diversity of solutions. Push and pull search (PPS) [31] divided the search process into two stages: push and pull search, and embedded the MOEA/D algorithm [29] into the PPS framework for tackling CMOPs. CCMO [32] used a coevolutionary framework of two populations to share information with each other for dealing with CMOPs. MOEA/D-DAE [33] developed a detect-and-escape strategy to avoid being trapped into local optima and struck in an unfeasible area.

Following the above ideas, we propose and compare three constrained multi-objective evolutionary algorithms (CMOEAs) to solve constrained multi-objective computation offloading problems in the collaborative edge-cloud computing environment. The major contributions of this paper are summarized as follows:

- Three constrained multi-objective evolutionary algorithms, i.e., PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE are developed by taking advantage of PPS framework and NSGA-II, PSEA2 and SPEA2-SDE with constraint handling principles.
- Three challenging constrained benchmark suites are selected to evaluate the performance of the three proposed algorithms, which are compared with the other seven state-of-the-art CMOEAs. The numerical results verify the effectiveness and competitiveness of the proposed algorithms.
- We further compare the three proposed algorithms with the other five representative CMOEAs as well as four offloading schemes to solve different scale computation offloading problems. In addition, impacts of different parameters in edge-cloud networks and different types of applications are analyzed with regard to the performance of different offloading policies. The experimental results demonstrate the superiority and efficiency of the proposed algorithms.

The remainder of this paper is organized as follows. Section II describes the background of the constrained multi-objective optimization and PPS framework. The system model and problem formulation are provided in Section III. The details of the three proposed algorithms are illustrated in Section IV. The simulation studies on benchmark suites are presented in Supplementary Materials I. The experimental results on computation offloading problems are discussed in Section V. Finally, Section VI draws the conclusion and future work.

II. Background

In this section, we introduce some concepts of constrained multi-objective optimization (CMOPs) and push and pull search (PPS) framework for solving CMOPs. PPS framework has been demonstrated to be a very efficient technology for dealing with CMOPs [31]. We try to apply the PPS framework to NSGA-II, SPEA2 and SPEA2-SDE for solving constrained multi-objective computation offloading problems. For better understanding the implementation process of PPS framework, we present the operation details of the push and pull stages as well as introduce the condition when to switch from the push to the pull search process. In addition, We illustrate the reason that PPS framework is a potential technique for solving CMOPs.

A. Constrained Multi-objective Optimization

Many real-world problems can be formulated as constrained multi-objective optimization problems (CMOPs), which aim to optimize different conflicting objectives simultaneously with a set of inequality and/or equality constraints. A CMOP can be defined as follows [31]:

$$\min: \ F(x) = \left(f_1(x), f_2(x), \cdots, f_m(x)\right)^T, \tag{1}$$

$$s.t.: \ \mathrm{g}_i(x) \geq 0, \ \ i = 1, \cdots, p, \tag{2}$$

$$h_j(x) = 0, \ \ j = 1, \cdots, q, \tag{3}$$

$$x = (x_1, x_2, \cdots, x_D) \in \Omega, \tag{4}$$

where x is a solution consisting of D decision variables, $\Omega \subseteq \mathbb{R}^D$ is the decision space, $F(x) \subseteq \mathbb{R}^m$ is an m-dimensional objective vector, $\mathrm{g}_i(x) \geq 0$ is an inequality constraint, $h_j(x) = 0$ is an equality constraint, and the number of inequality and equality constraints are p and q, respectively.

When solving CMOPs with equality constraints, we often relax the equality constraint with an extremely small positive value δ and convert the equality constraints into inequality constraints, which can be expressed as:

$$h_j(x)' \equiv \delta - |h_j(x)| \geq 0. \tag{5}$$

In order to deal with CMOPs with different inequality and quality constraints, the overall constraint violation of each solution x can be calculated as:

$$CV(x) = \sum_{i=1}^{p} |\min\{\mathrm{g}_i(x), 0\}| + \sum_{i=1}^{q} \left|\min\left\{h_j(x)', 0\right\}\right|, \tag{6}$$

where x is a feasible solution if $CV(x) = 0$, otherwise it is infeasible. A feasible solution x^a is said to Pareto dominate another feasible solution x^b, denoted by $x^a \prec x^b$, if every objective value of x^a is not greater than that value of x^b and there exists at least one objective value of x^a is less than x^b. If there are no other feasible solutions dominating solution x^*, which is called a Pareto optimal solution. All the Pareto optimal solutions constitute the Pareto optimal set (PS). And the mapping of a Pareto optimal set into the objective space is called Pareto front (PF).

B. PPS Framework

The push and pull search (PPS) framework was proposed to solve CMOPs by Fan et al. [31]. The search process of PPS is divided into two different stages: push and pull search stages. In the first push stage, the working population is pushed to approach the unconstrained Pareto front without considering any constraints, which can help the solutions to get across infeasible regions. Afterwards, a constraint handling mechanism is used to pull the working population to approach the constrained Pareto front in the pull stage.

The condition when to convert from the push stage to pull stage is important, which can be suggested as [31], [34]:

$$r_k = \max\{rz_k, rn_k\} \leq \varepsilon, \tag{7}$$

where ε (suggested $\varepsilon = 0.001$) is a threshold. r_k denotes the maximum rate of change between the ideal and nadir points during the last l generations. rz_k and rn_k represent the rates of change of the ideal and nadir points during the last l generations, defined as follows:

$$rz_k = \max_{i=1,\cdots,m} \left\{ \frac{\left|z_i^k - z_i^{k-l}\right|}{\max\left\{\left|z_i^{k-l}\right|, \Delta\right\}} \right\}, \tag{8}$$

$$rn_k = \max_{i=1,\cdots,m} \left\{ \frac{\left|n_i^k - n_i^{k-l}\right|}{\max\left\{\left|n_i^{k-l}\right|, \Delta\right\}} \right\}, \tag{9}$$

where $z^k = \left(z_1^k, \cdots, z_m^k\right)$ and $n^k = \left(n_1^k, \cdots, n_m^k\right)$ are the ideal and nadir points in the k-th generation, respectively. $z^{k-l} = \left(z_1^{k-l}, \cdots, z_m^{k-l}\right)$ and $n^{k-l} = \left(n_1^{k-l}, \cdots, n_m^{k-l}\right)$ are the ideal and nadir points in the $k-l$-th generation. Δ (suggested $\Delta = 1e{-}6$) is a very small positive number, which is used to make sure that the denominators in Eqs. (8) and (9) are not equal to zero. rz_k and rn_k are two points in the interval [0,1].

r_k is initialized 1 at the beginning of the search, and is updated at each iteration according to Eq. (7). When r_k is less than or equal to ε, the push stage will be transformed into pull stage.

To summarize, PPS has two potential advantages over other constraint handling techniques [34]. During the first push stage, a multi-objective evolutionary algorithm is adopted to approximate the Pareto front without considering any constraints, which can help the working population to get across the large infeasible regions and avoid the distance between the unconstrained PF and true PF. After obtaining the unconstrained PF in the push stage, some valuable information can be collected to guide the parameter setting for the constraint handling approaches in the pull stage, which can enhance the adaptability of the algorithm.

III. System Model and Problem Formulation

In this section, we consider a collaborative MEC and MCC network with multiple mobile devices (MDs), multiple edge servers and multiple cloud servers. The computation tasks in the mobile devices can be executed locally or offloaded to the edge/cloud servers.

A. System Model

Fig. 1 presents the system model composed by L cloud servers, K edge servers, and N mobile devices (MDs). Each MD can communicate with the edge server with a wireless link, whereas the edge server and cloud server are connected through a wired link. Without loss of generality, we assume that each mobile device has M independent tasks. We denote the set of MDs as $\mathcal{N} = \{1, 2, \cdots, N\}$ and the set of tasks as $\mathcal{M} = \{1, 2, \cdots, M\}$, and the set of servers as $\mathcal{K} = \{0, 1, 2, \cdots, K, K+1, \cdots, K+L\}$, where server 0 denotes MD itself and servers $\{1, 2, \cdots, K\}$ denote the edge servers and servers $\{K+1, \cdots, K+L\}$ denote the cloud servers. In each MD, different tasks can decide to be processed by MD itself or remotely processed by edge/cloud servers. We denote $a_{nm} \in \{0, 1, 2, \cdots, K, K+1, \cdots, K+L\}$ as the offloading decision that MD n's m-th task is assigned to mobile device or cloud/edge servers, where $n \in \mathcal{N}$ and $M \in \mathcal{M}$. Especially, $a_{nm} = 0$ means that MD n chooses to locally execute its m-th task, $a_{nm} \in \{1, 2, \cdots, K\}$ indicates that MD n's m-th task is offloaded to the edge servers and $a_{nm} \in \{K+1, K+2, \cdots, K+L\}$ represents that MD n's m-th task is offloaded to the cloud servers. Overall, every task must be processed locally or by the edge/cloud servers, whose offloading decision depends on:

$$a_{nm} = \begin{cases} 0, & \text{local computing,} \\ \in \{1, 2, \cdots, K\}, & \text{edge computing,} \\ \in \{K+1, K+2, \cdots, K+L\}, & \text{cloud computing.} \end{cases} \tag{10}$$

where $n \in \mathcal{N}$ and $M \in \mathcal{M}$. Since both response time and energy consumption play a significant role in the performance of computation offloading for MDs, we consider these two objectives as QoS metrics. The detailed operations of the communication and computation process are illustrated in Sections III-B and III-C, respectively. The important notations used in this paper are listed in Table I.

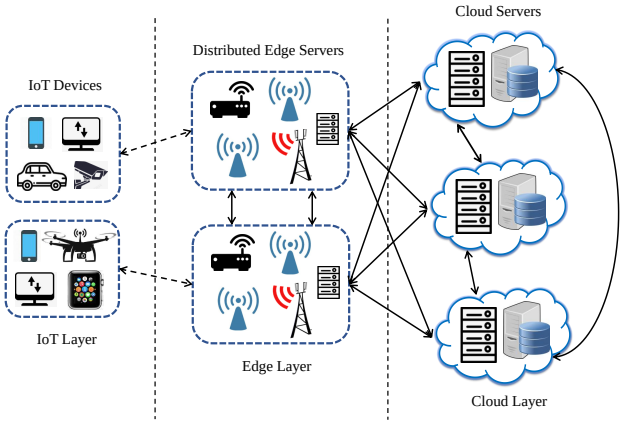


Fig. 1: System model of local-edge-cloud computation offloading.

B. Communication Model

Considering the communication cost between the MDs and edge/cloud servers, we first analyze the transmission time and energy consumption in the communication model. We set a tuple $(\alpha_{nm}, \gamma_{nm})$ to represent MD n's m-th task, where α_{nm} is the data size and γ_{nm} is the required number of CPU cycles to finish the task. When one of the MD n's task m is offloaded to the edge server $k \in \{1, 2, \cdots, K\}$, the whole processing of task m includes transmitting and edge computing phase. Let B_{nk}^{UE} denote the allocated upload bandwidth between the MD n and the edge server k. We neglect the influence of the process when the edge server returns the results back to MDs since the data size of feedback information is small in general [15]. The upload transmission time for offloading MD n's m-th task to the edge server k can be calculated as:

$$T_{nm}^{UE} = \frac{\alpha_{nm}}{B_{nk}^{UE}}. \tag{11}$$

The energy consumption for uploading MD n's m-th task to the edge server k can be quantified as:

$$E_{nm}^{UE} = P_n^{TX} T_{nm}^{UE}. \tag{12}$$

where P_n^{TX} is the transmission energy consumption power of the MD n.

When one of the MD n's task is offloaded to the cloud server $k \in \{K+1, K+2, \cdots, K+L\}$, one of the edge servers is selected as a relay node between the MD and the cloud server. We assume that the task is first transmitted to the edge server $\widetilde{k}$ through a wireless link, then the edge server $\widetilde{k}$ will forward the task to the central cloud server k via a wired link. The upload transmission time for offloading MD n's m-th task to the cloud server k can be calculated as:

$$T_{nm}^{UC} = \frac{\alpha_{nm}}{B_{n\widetilde{k}}^{UE}} + \tau, \tag{13}$$

where τ denotes the propagation delay between edge servers and cloud servers. We focus on the energy consumption of MDs, thus the energy consumption for uploading MD n's m-th task to the cloud server k can be quantified as:

$$E_{nm}^{UC} = P_n^{TX} \times \left(T_{nm}^{UC} - \tau\right). \tag{14}$$

When the task is executed locally, there is no communication latency. Hence, the total communication delay of MD n for completing all M tasks can be expressed as:

$$T_n^{Comm} = T_n^{CommE} + T_n^{CommC}, \tag{15}$$

where

$$\begin{cases} T_n^{CommE} = \sum_{m=1}^{M} T_{nm}^{UE}, \ a_{nm} \in \{1, 2, \cdots, K\}, \\ T_n^{CommC} = \sum_{m=1}^{M} T_{nm}^{UC}, \ a_{nm} \in \{K+1, \cdots, K+L\}. \end{cases} \tag{16}$$

Then the overall communication energy consumption of MD n for completing all M tasks can be calculated as:

$$E_n^{Comm} = E_n^{CommE} + E_n^{CommC}, \tag{17}$$

where

$$\begin{cases} E_n^{CommE} = \sum_{m=1}^{M} E_{nm}^{UE}, \ a_{nm} \in \{1, 2, \cdots, K\}, \\ E_n^{CommC} = \sum_{m=1}^{M} E_{nm}^{UC}, \ a_{nm} \in \{K+1, \cdots, K+L\}. \end{cases} \tag{18}$$

TABLE I: Important notations used in this paper.

Notation	Description
a_{nm}	The offloading decision of m-task of n-th MD
α_{nm}	Input data size of the task m of MD n
γ_{nm}	Total CPU cycles of the task m of MD n
B_{nk}^{UE}	The transmission bandwidth between MD n and edge server k
P_n^{TX}	The transmission power consumption of MD n
T_{nm}^{UE}	The transmission time for offloading task m of MD n to edge server k
E_{nm}^{UE}	The transmission energy consumption for offloading task m of MD n to edge server k
τ	The propagation latency between a edge server and a cloud server
T_n^{CommE}	The transmission latency from MD n to edge servers
T_n^{CommC}	The transmission latency from MD n to cloud servers
T_n^{Comm}	The total communication delay of MD n for completing all M tasks
E_n^{CommE}	The communication energy consumption from MD n to edge servers
E_n^{CommC}	The communication energy consumption from MD n to cloud servers
E_n^{Comm}	The total communication energy consumption of MD n for completing all M tasks
f_l, f_e, f_c	The CPU frequency in mobile devices, edge servers and cloud servers
T_{nm}^{Comp}	The computation latency of m task of MD n
T_n^{CompL}	The total computation latency of MD n in mobile devices
T_n^{CompE}	The total computation latency of MD n in edge servers
T_n^{CompC}	The total computation latency of MD n in cloud servers
E_n^{Comp}	The total computation energy consumption of MD n
T_n	The overall completion time of executing all M tasks of MD n
T	The overall completion time of executing all tasks of all MDs
E_n	The energy consumption of executing all M tasks of MD n
E	The total energy consumption of executing all tasks of all MDs
T^{Cons}	The response time constraint
E^{Cons}	The energy consumption constraint

C. Computation Model

We denote f_l, f_e, f_c as the number of CPU cycles for the mobile devices, the edge servers and the cloud servers, respectively. In general, the computation capability of the cloud servers is more powerful than the edge servers, and the edge servers have better computation capability than the mobile devices, as $f_l \ll f_e \ll f_c$.

When each task is determined to be offloaded to edge or cloud servers, the edge or cloud servers start to process it after all the input data has been received by the edge or cloud servers. The computation latency of MD n's m-th task in MDs, the edge servers and cloud servers are calculated as:

$$T_{nm}^{Comp} = \begin{cases} \frac{\gamma_{nm}}{f_l}, & a_{nm} = 0, \\ \frac{\gamma_{nm}}{f_e}, & a_{nm} \in \{1, 2, \cdots, K\}, \\ \frac{\gamma_{nm}}{f_c}, & a_{nm} \in \{K+1, \cdots, K+L\}. \end{cases} \tag{19}$$

Thus, the total computation latency of MD n for completing all M tasks can be expressed as:

$$\begin{cases} T_n^{CompL} = \sum\limits_{m=1}^{M} \frac{\gamma_{nm}}{f_l}, a_{nm} = 0, \\ T_n^{CompE} = \sum\limits_{m=1}^{M} \frac{\gamma_{nm}}{f_e}, a_{nm} \in \{1, 2, \cdots, K\}, \\ T_n^{CompC} = \sum\limits_{m=1}^{M} \frac{\gamma_{nm}}{f_c}, a_{nm} \in \{K+1, \cdots, K+L\}. \end{cases} \tag{20}$$

In this paper, we only consider the energy consumption at MDs. Specially, we use P_n^L to denote the local energy consumption power of MD n. Then MD n's energy consumption for executing its task m locally is given by:

$$E_{nm}^{Comp} = P_n^L \times \frac{\gamma_{nm}}{f_l}. \tag{21}$$

Hence, the total computation energy consumption of MD n can be expressed as:

$$E_n^{Comp} = P_n^L \times T_n^{CompL}. \tag{22}$$

D. Problem Formulation

The processing latency consists of communication and computation latency, and the total delay of executing all M tasks of MD n can be given by:

$$T_n = \max\{T_n^{CompL}, T_n^{CompE} + T_n^{CommE}, T_n^{CompC} + T_n^{CommC}\}. \tag{23}$$

The total completion time of executing all tasks of all MDs can be expressed:

$$T = \max\left\{ \sum_{n=1}^{N} T_n^{CompL}, \sum_{n=1}^{N} \left(T_n^{CompE} + T_n^{CommE}\right), \sum_{n=1}^{N} \left(T_n^{CompC} + T_n^{CommC}\right) \right\}. \tag{24}$$

The energy consumption of executing all M tasks of MD n can be given by:

$$E_n = E_n^{Comp} + E_n^{Comm}. \tag{25}$$

The total energy consumption of executing all tasks of all MDs can be expressed as:

$$E = \sum_{n=1}^{N} \left(E_n^{Comp} + E_n^{Comm}\right). \tag{26}$$

Hence, the computation offloading problem can be formalized as follows:

$$\min: \ [T, E], \tag{27}$$

$$s.t.: \ a_{nm} \in \{0, 1, 2, \cdots, K, K+1, \cdots, K+L\}, \tag{28}$$

$$|a_{nm}| = 1, \tag{29}$$

$$T \leq T^{\text{Cons}}, \tag{30}$$

$$E \leq E^{\text{Cons}}, \tag{31}$$

where the first and second constraints indicate that each task is assigned to one server, the third constraint denotes that MDs have constraints of response time deadline, and the last constraint represents the energy consumption limits. To summarize, we establish a local-edge-cloud constrained multi-objective computation offloading model.

IV. The Proposed Algorithms

This section presents the details of three proposed algorithms PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE.

A. The General Framework

The general frameworks of the three proposed algorithms are presented in Algorithm 1 and Algorithm 2, respectively. The flowchart of PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE is shown in Fig. 2. Please note that PPS-SPEA2 and PPS-SPEA2-SDE share the same framework but have different fitness calculating methods. In the framework of PPS-NSGA-II, the non-dominated front numbers and crowding distances of solutions are calculated by the fast non-dominated sorting approach [27] with and without considering constraints, respectively. The whole search process consists of two stages: push and pull search. When $PushStage = true$, the push stage is utilized, the parents are selected via binary tournament selection as the mating pool without considering constraints and then offspring solutions O are generated from the mating pool. When $PushStage = false$, the pull stage is applied, a constraint handling mechanism is embedded into NSGA-II to pull the working population to the constrained PF. The parameter r_k for switching from push to pull stage is updated iteratively.

In the framework of PPS-SPEA2 and PPS-SPEA2-SDE, different fitness calculation methods are adopted instead of calculating the non-dominated front numbers and crowding distances in PPS-NSGA-II. The fitness calculating methods can reflect both the performance of convergence and diversity of each solution in the population. Without loss of generality, the smaller fitness value means better performance. The whole search processes of PPS-SPEA2 and PPS-SPEA2-SDE also include push and pull search stages. In the push stage, we use SPEA2 and SPEA2-SDE without considering any constraints to search the unconstrained PF. In the pull stage, the constraint handing approaches are applied to search the constrained PF. More details about the main operations in PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE are presented in Sections IV-B, IV-C, and IV-D, respectively.

Algorithm 1 Framework of PPS-NSGA-II

Input: The population size $\widetilde{N}$
Output: The final population P

$P \leftarrow Initialization\left(\widetilde{N}\right)$;
$[F_1, F_2, \cdots] \leftarrow NDSorting\left(P.objs\right)$;
$CrowdDis \leftarrow CrowdingDistance\left(F_1, F_2, \cdots\right)$;
$\left[F_1{}', F_2{}', \cdots\right] \leftarrow NDSorting\left(P.objs, P.cons\right)$;
$CrowdDis' \leftarrow CrowdingDistance\left(F_1{}', F_2{}', \cdots\right)$;
Set $r_k = 1.0$, $PushStage = true$;
while $termination\ criterion\ not\ fulfilled$ **do**
  Calculate r_k according to Eq. (7);
  if $r_k \leq \varepsilon$ and $PushStage = true$ **then**
    $PushStage = false$;
  end if
  if $PushStage = true$ **then**
    $P' \leftarrow$ Select $\widetilde{N}$ parents via binary tournament selection according to $[F_1, F_2, \cdots]$ and $CrowdDis$ in P;
    $O \leftarrow OffspringGeneration\left(P, P'\right)$;
    $\left(P, [F_1, F_2, \cdots], CrowdDis\right) \leftarrow EnvironmentalSelection\left(P \cup O\right)$;
  else
    $P' \leftarrow$ Select $\widetilde{N}$ parents via binary tournament selection according to $\left[F_1{}', F_2{}', \cdots\right]$ and $CrowdDis'$ in P;
    $O \leftarrow OffspringGeneration\left(P, P'\right)$;
    $\left(P, \left[F_1{}', F_2{}', \cdots\right], CrowdDis'\right) \leftarrow EnvironmentalSelection'\left(P \cup O\right)$;
  end if
end while

B. PPS-NSGA-II

In the original NSGA-II, Deb et al. [27] embedded feasibility into Pareto dominance and developed a constraint dominance principle (CDP) to deal with constraints. If a solution x^i is said to constrained-dominate a solution x^j, one of the following three conditions holds: 1) x^i is a feasible solution and x^j is an infeasible solution. 2) Solutions x^i and x^j are both feasible solutions, and solution x^i Pareto dominates solution x^j in terms of objectives. 3) Solutions x^i and x^j are both infeasible solutions, and solution x^i has a lower overall constraint violation than that of solution x^j.

PPS-NSGA-II is an instantiation of the PPS framework of a specific type of NSGA-II algorithm [27]. In the push search stage, we use an unconstrained NSGA-II to search for both feasible and infeasible solutions to minimize the objectives of solutions without considering any constraints, which aims to approach the unconstrained PF. The non-dominated front numbers and crowding distances of solutions are calculated by the fast non-dominated sorting approach. The crowding distance is defined as the average distance between its two closest points on each objective. Then $\widetilde{N}$ parents are selected as mating pool via binary tournament selection based on the non-dominated front numbers and crowding distances. The two parents are randomly selected from the mating pool to generate two offspring solutions, and a genetic operator [27] or

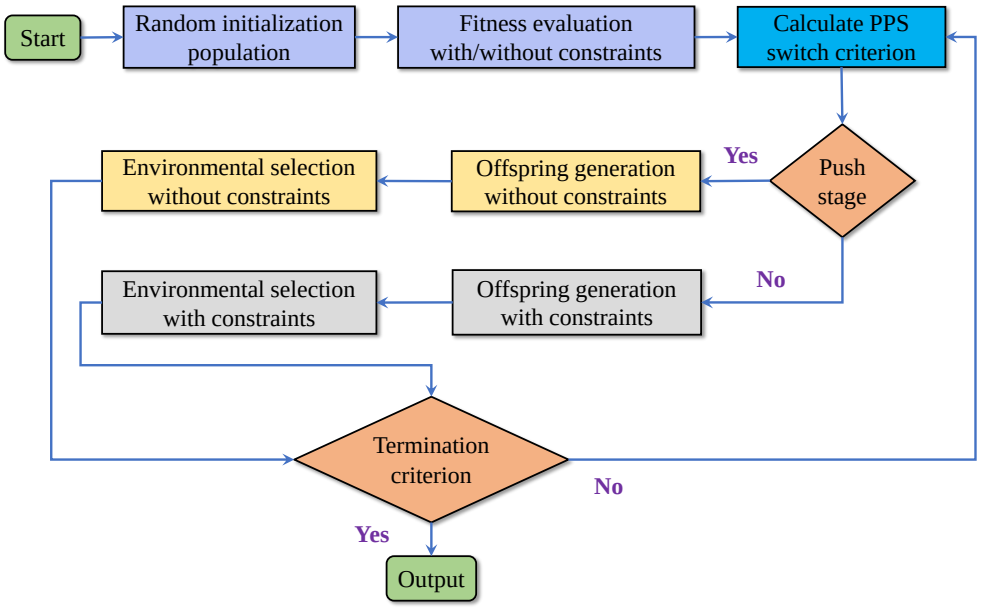


Fig. 2: The flowchart of PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE.

Algorithm 2 Frameworks of PPS-SPEA2 and PPS-SPEA2-SDE

Input: The population size $\widetilde{N}$
Output: The final population P

$P \leftarrow Initialization\left(\widetilde{N}\right)$;
$Fitness \leftarrow CalFitness\left(P.objs\right)$;
$Fitness' \leftarrow CalFitness\left(P.objs, P.cons\right)$;
Set $r_k = 1.0$, $PushStage = true$;
while *termination criterion not fulfilled* **do**
  Calculate r_k according to Eq. (7);
  if $r_k \leq \varepsilon$ and $PushStage = true$ **then**
    $PushStage = false$;
  end if
  if $PushStage = true$ **then**
    $P' \leftarrow$ Select $\widetilde{N}$ parents via binary tournament selection according to $Fitness$ in P;
    $O \leftarrow OffspringGeneration\left(P, P'\right)$;
    $\left(P, Fitness\right) \leftarrow EnvironmentalSelection\left(P \cup O\right)$;
  else
    $P' \leftarrow$ Select $\widetilde{N}$ parents via binary tournament selection according to $Fitness'$ in P;
    $O \leftarrow OffspringGeneration\left(P, P'\right)$;
    $\left(P, Fitness'\right) \leftarrow EnvironmentalSelection'\left(P \cup O\right)$;
  end if
end while

differential evolution operator [35] can be applied as offspring generating operator. Thus the environmental selection operation is adopted to update the non-dominated front numbers and crowding distances as well as the new population.

The ideal and nadir points are updated at each iteration. And the maximum rate of change between the ideal and nadir points (r_k) during the last l generations is calculated. When r_k satisfies the condition of switching from the push to pull stages, the pull search stage is starting. In the pull search stage, the constraint dominance principle (CDP) is applied to calculate the non-dominated front numbers and crowding distances. Then the new mating pool and offspring solutions are generated based on the new non-dominated front numbers and crowding distances with respect to the constraints. Finally, a set of feasible solutions will be updated and obtained in the environmental selection operation.

C. PPS-SPEA2

PPS-SPEA2 is an instantiation of the PPS framework of a specific type of SPEA2 algorithm [36]. In PPS-NSGA-II, the non-dominated front number represents the performance of convergence and the crowding distance reflects the performance of diversity. However, the fitness metric value is used to measure both convergence and diversity in PPS-SPEA2. The fitness evaluation strategy shares the same idea as the one in the original SPEA2. First of all, let the solution set R_x store all the solutions dominated by x and the solution set S_x store all the solutions dominating x, the raw fitness $R\left(x\right)$ of a solution x is calculated as:

$$R\left(x\right) = \sum_{y \in S_x} |R_y|, \tag{32}$$

where $|R_y|$ denotes the number of solutions in the set. $R\left(x\right) = 0$ means solution x is a non-dominated solution. What's more, additional density information is needed to distinguish the quality of different non-dominated solutions. The k-th nearest neighbor method [37] is applied to measure the density information of solutions. Then $\left\lfloor \sqrt{2\widetilde{N}} \right\rfloor$-th nearest neighbor x' of solution x is detected, the density $D\left(x\right)$ of corresponding to x is calculated as:

$$D\left(x\right) = \frac{1}{dist\left(x, x'\right) + 2}, \tag{33}$$

where $dist(x, x')$ denotes the Euclidean distance between solutions x and x'.

Hence, the fitness of the solution x can be expressed as follows:

$$fit(x) = R(x) + D(x), \tag{34}$$

where x is the non-dominated solution when $fit(x) < 1$. Obviously, smaller fitness means better quality of the solution.

PPS-SPEA2 also has two search stages: push and pull stages. In the push stage, no constraints will be considered into the fitness evaluation method, PPS-SPEA2 can search for unconstrained solutions. In the pull stage, the constraint dominance principle (CDP) is embedded into the fitness evaluation method, PPS-SPEA2 can pull the unconstrained solutions to the feasible regions. It is necessary to point out that the solution which has the minimum distance to another solution is chosen to be deleted in the environmental selection operation. If there are several solutions having the same minimum distance, we consider the second smallest distances and so forth.

D. PPS-SPEA2-SDE

PPS-SPEA2-SDE is an instantiation of the PPS framework of a specific type of SPEA2-SDE algorithm [38]. Compared PPS-SPEA2 with PPS-SPEA2-SDE, the fitness calculating method is different. In PPS-SPEA2-SDE, the shift-based density estimation (SDE) strategy is used to measure the density of the solutions. The shifted-based density estimation based distance between solution x and solution y ($y \in P \backslash \{x\}$) can be calculated as:

$$SDE(x, y) = \sqrt{\sum_{i=1}^{m} \left(\max\{0, f_i(y) - f_i(x)\}\right)^2}. \tag{35}$$

Similar to PPS-SPEA2, the fitness of the solution x can be expressed as follows:

$$fit(x) = R(x) + \frac{1}{SDE(x, x') + 2}, \tag{36}$$

where $R(x)$ is the same to that of PPS-SPEA2. $SDE(x, x')$ is the SDE crowding degree of the solution x with regard to its $\left\lfloor \sqrt{2\widetilde{N}} \right\rfloor$-th nearest neighbor x'. Afterwards, PPS-SPEA2-SDE shares the same search process with PPS-SPEA2. It is noted that the solution which has the minimum SDE based distance to another solution is chosen to be deleted in the environmental selection operation. If there are several solutions having the same minimum SDE based distance, we consider the second smallest distances and so forth.

E. Computational Complexity

For the proposed algorithms PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE, the major costs are the iteration process in Algorithm 1 and 2. In PPS-NSGA-II, the worst-case time complexities of the maximum rate of change between the ideal and nadir points are $O(\widetilde{M}\widetilde{N})$, where $\widetilde{M}$ is the number of objectives and $\widetilde{N}$ is the population size. The mating selection operator needs $O(\widetilde{N})$ operations for the binary tournament selection. The offspring reproduction needs $O(\widetilde{N}D)$ operations to generate offspring solutions, where D is the number of decision variables. The non-dominated sorting operator and environmental selection operator need $O(\widetilde{M}\widetilde{N}^2)$ operations. Thus, the overall computational complexity of PPS-NSGA-II within one generation is $O(\widetilde{M}\widetilde{N}^2)$.

In PPS-SPEA2, the time complexity of the fitness calculating procedure is $O(\widetilde{N}^2 \log \widetilde{N})$. The binary tournament selection needs $O(\widetilde{N})$ operations and the offspring generation needs $O(\widetilde{N}D)$ operations. The worst run-time complexity of the environmental selection operator is $O(\widetilde{N}^3)$, on average the complexity will be lower $O(\widetilde{N}^2 \log \widetilde{N})$ [36]. Thus, the worst overall computational complexity of PPS-SPEA2 within one generation is $O(\widetilde{N}^3)$. In PPS-SPEA2-SDE, the time complexity of the fitness calculating procedure is $O(\widetilde{M}\widetilde{N}^2)$. And the worst run-time complexity of the environmental selection operator is $O(\widetilde{N}^3)$. Since $\widetilde{N}$ is often larger than $\widetilde{M}$. Hence, the worst overall computational complexity of PPS-SPEA2-SDE within one generation is $O(\widetilde{N}^3)$.

V. Performance Evaluation

Before using the proposed algorithms to deal with offloading problems, we adopt three challenging benchmark suites to test the performance and the simulation results are illustrated in Supplementary Materials I. In this section, we further study the performance of PPS-NSGA-II, PPS-SPEA2, PPS-SPEA2-SDE for solving constrained multi-objective computation offloading optimization problems.

A. Experimental Setup

We set up the multi-server multi-user multi-task computation offloading scenario in the local-edge-cloud environment. The number of mobile devices is selected between 10 and 100. The number of independent tasks of each MD is $M = 5$. We set the number of edge servers $K = 5$ and the number of cloud serves $L = 2$. In the following scenarios, we consider the CPU frequencies of each MD, each edge server, and each cloud server are 0.6 GHz, 10 GHz and 1 THz, respectively [12]. The transmitting power P_n^{TX} of all MDs is 0.2 W. The power consumption of all MDs is 0.7 W. The round-trip propagation delay between edge servers and cloud servers is $\tau = 15$ ms. The bandwidth between MDs and edges is randomly selected from [8,15] MBps. The data size of each task is uniformly distributed between 10 MB and 30 MB. The total CPU cycles for finishing the task are assumed to be proportional to the input data size [3], i.e., $\gamma_{nm} = \rho\alpha_{nm}$. Here the parameter ρ denotes the computation to data radio for different types of applications. Table II lists some values of ρ for various applications [39], [40]. For example, the label A represents the gzip application and $\rho = 330$ cycles/byte. By default, the type A application is taken as an example of the computation offloading problems.

To verify the performance of the proposed algorithms, we compare PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE with other five algorithms TiGE-2 [41], constrained NSGA-II [27], PPS-MOEA/D [31], ToP [42] and CMOEA-MS [43] to solve five offloading problems, which consider that the number of MDs N = [10, 30, 50, 70, 100]. For a fair comparison, the

TABLE II: Application complexity.

Application	Labels	ρ (cycles/byte)
gzip	A	330
pdf2text (N900 data sheet)	B	960
x264 CBR encode	C	1900
html2text	D	5900
pdf2text (E72 data sheet)	E	8900

population size of all algorithms is set to 100, and the number of iterations is 1000. The solution encoding style adopts the real-encoding method, which means that each task is assigned to a specific server including edge and cloud servers. We apply the hypervolume (HV) [44] as the performance metric to evaluate the performance of these compared algorithms. Each algorithm is executed 30 times independently on each test problem, and the average and standard deviation of performance metric values are recorded. The Wilcoxon rank-sum test at a 5% significance level is used to compare the experimental results, where the symbol '$+$', '$-$' and '$\approx$' denotes that the result of another algorithm is significantly better, significantly worse and similar to that obtained by the proposed algorithm.

B. Convergence Properties Analysis

As listed in Table III, the proposed PPS-NSGA-II has achieved the best performance on four offloading problems, while only CMOEA-MS gets one best result among other algorithms. It is necessary to point out that CMOEA-MS, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE share similar overall performance based on the Wilcoxon rank-sum test, which outperform other four compared algorithms (TiGE-2, NSGA-II, PPS-MOEA/D and ToP). We can also observe that PPS-MOEA/D may obtain good performance for solving the benchmark suites, while encountering difficulties in solving discrete computation offloading problems.

We can observe that ToP cannot find any feasible solutions on N = 50 and 100 offloading problems as shown in Figs. 3(b) and 3(c). TiGE-2, NSGA-II, and PPS-MOEA/D can obtain a few feasible and non-dominated solutions. NSGA-II, CMOEA-MS, PPS-SPEA2 and PPS-SPEA2-SDE may get good results about the small-scale offloading problems (e.g., N = 10), while their performance deteriorates with the growth of the number of mobile devices, especially for the algorithm NSGA-II. PPS-NSGA-II can always obtain a set of well-distributed and well-converged feasible solutions for different offloading problems.

C. Performance of Different Offloading Schemes

It has been demonstrated that PPS-NSGA-II has a good and stable performance in terms of both convergence and diversity on different offloading problems. To further evaluate the performance of PPS-NSGA-II for reducing response time and energy consumption, we compare PPS-NSGA-II with other four offloading schemes, which are Local Offloading Scheme (LOS), Edge Offloading Scheme (EOS), Cloud Offloading Scheme (COS), and Random Offloading Scheme (ROS). LOS, EOS, and COS represent that all tasks are executed locally, offloaded to edge servers and central cloud servers. ROS denotes that offloading decisions of all tasks are generated randomly. In order to better compare the effectiveness of different algorithms, we can design system cost and offloading gain of a weighted sum of time and energy as follows:

$$SystemCost = w \times T_{offloading} + (1-w) \times E_{offloading}, \tag{37}$$

$$OffloadingGain = \left[w \times \frac{T_{LOS} - T_{offloading}}{T_{LOS}} + (1-w) \times \frac{E_{LOS} - E_{offloading}}{E_{LOS}} \right] \times 100\%, \tag{38}$$

where $T_{offloading}$ and $E_{offloading}$ denote overall time and energy consumption of one specific offloading scheme, respectively. T_{LOS} and E_{LOS} denote the time and energy consumption of LOS, respectively. w is the weight trade-off parameter between time and energy, which can be set by the decision-maker. The larger w is, the more sensitive the response time is.

Figs. 4, 5, and 6 present the offloading gain of different offloading schemes under different weights. Compared with LOS, all the other offloading schemes benefit a lot with regard to time consumption and energy consumption. PPS-NSGA-II can obtain the best offloading gain compared with other offloading schemes among all the different offloading problems with different weights. COS achieves a better offloading gain performance than EOS since the cloud servers take obvious advantages of powerful cloud resources over edge servers. It is noted that EOS gains better results compared with ROS when w = 0.2 and 0.5, while ROS may outperform EOS in the case w = 0.8 (focus on time consumption) because the cloud server's powerful computing capability achieves high response time efficiency performance.

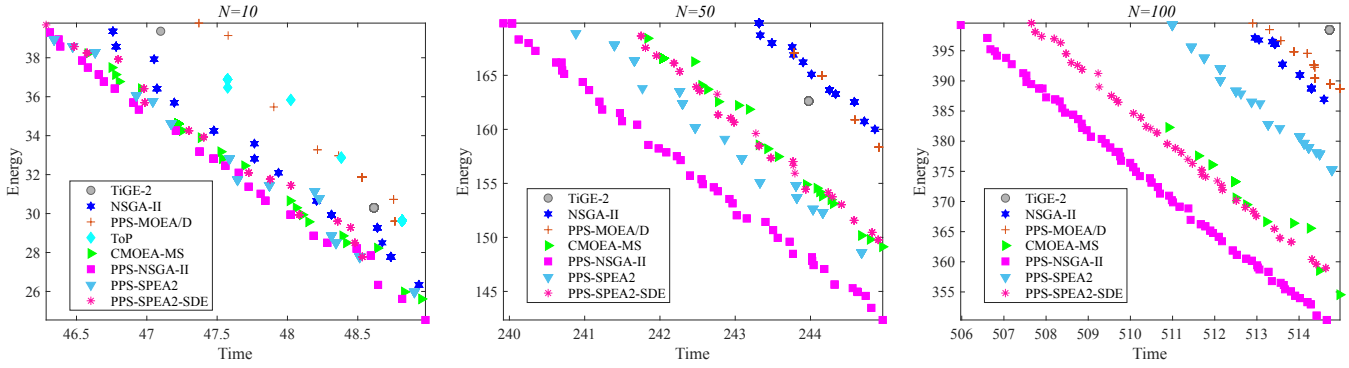
D. Impacts of Different Parameters

In this section, we analyze the impacts of different parameters in collaborative edge-cloud computing networks, and w is set to 0.5 as well as N is equal to 10. Fig. 7 illustrates the performance of system cost and offloading gain on different offloading schemes under the different number of tasks of each MD. PPS-NSGA-II gains the best performance compared with other offloading schemes. With the increasing number of tasks, the system cost of LOS grows much faster than EOS, COS, ROS, and PPS-NSGA-II. The offloading gain of the different offloading schemes stays stable since the system cost of EOS, COS, ROS, and PPS-NSGA-II belongs to a small relevant proportion of LOS.

Fig. 8 shows the performance of system cost and offloading gain on different offloading schemes under the different wireless bandwidth between MDs and edge servers. LOS does not change with the increment of wireless bandwidth. Both the performance of the system cost as well as offloading gain of the other four offloading schemes (EOS, COS, ROS, and PPS-NSGA-II) improve due to larger wireless bandwidth. In addition, with the increment of wireless bandwidth, the performance improves very fast at the beginning and then becomes small. It is worth noting that the offloading gain of

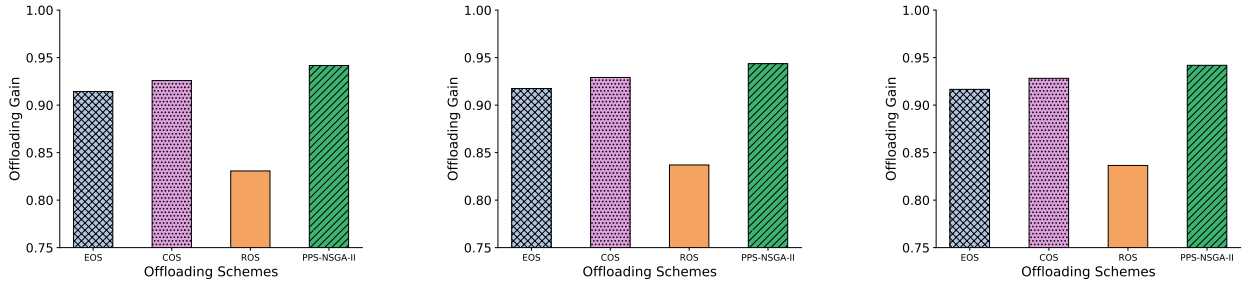
TABLE III: The HV values obtained by TiGE-2, NSGA-II, PPS-MOEA/D, ToP, CMOEA-MS, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE on five offloading problems. The best result in each row is highlighted. 'N/A' indicates that no feasible solution is found.

Problem	N	TiGE-2	NSGA-II	PPS-MOEA/D	ToP
Offloading1	10	3.2142e-1 (1.60e-2) −	3.4539e-1 (9.90e-3) −	3.2459e-1 (8.59e-3) −	2.8488e-1 (2.14e-2) −
Offloading2	30	2.7354e-1 (6.73e-3) −	1.5740e-1 (1.36e-1) −	7.6545e-2 (1.23e-1) −	5.1039e-2 (1.08e-1) −
Offloading3	50	2.6492e-1 (7.76e-3) −	1.8942e-1 (1.31e-1) −	1.0244e-1 (1.32e-1) −	N/A
Offloading4	70	2.4061e-1 (8.49e-2) −	2.4333e-1 (8.63e-2) −	1.2822e-1 (1.35e-1) −	5.0319e-2 (1.06e-1) −
Offloading5	100	1.8015e-1 (1.24e-1) −	1.2960e-1 (1.37e-1) −	5.1053e-2 (1.08e-1) −	N/A
$+/-/\approx$		0/5/0	0/5/0	0/5/0	0/3/0
Problem	N	CMOEA-MS	PPS-NSGA-II	PPS-SPEA2	PPS-SPEA2-SDE
Offloading1	10	3.4966e-1 (9.82e-3) ≈	3.5209e-1 (7.42e-3) ≈	3.5051e-1 (7.43e-3) ≈	3.4799e-1 (7.17e-3)
Offloading2	30	2.9020e-1 (7.39e-3) ≈	2.8669e-1 (7.55e-3) ≈	2.8761e-1 (4.38e-3) ≈	2.8279e-1 (9.65e-2)
Offloading3	50	2.8118e-1 (1.17e-2) ≈	2.8837e-1 (5.51e-3) ≈	2.8482e-1 (8.87e-3) ≈	2.8182e-1 (1.18e-2)
Offloading4	70	2.8290e-1 (7.49e-3) ≈	2.8896e-1 (3.41e-3) ≈	2.8520e-1 (7.26e-3) ≈	2.8635e-1 (2.46e-3)
Offloading5	100	2.7111e-1 (9.13e-3) ≈	2.7197e-1 (1.11e-2) ≈	2.7020e-1 (1.35e-3) ≈	2.7055e-1 (5.87e-3)
$+/-/\approx$		0/0/5	0/0/5	0/0/5	



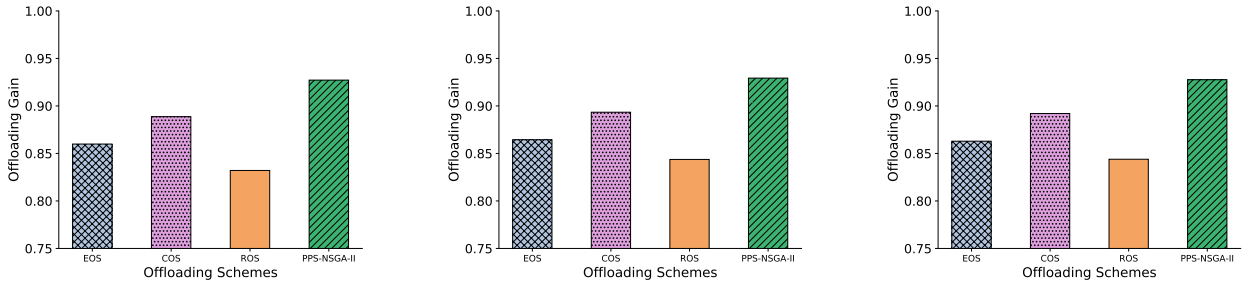
(a) $N = 10$ (b) $N = 50$ (c) $N = 100$

Fig. 3: The non-dominated solution sets with the medium HV value obtained by TiGE-2, NSGA-II, PPS-MOEA/D, ToP, CMOEA-MS, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE on different offloading problems.



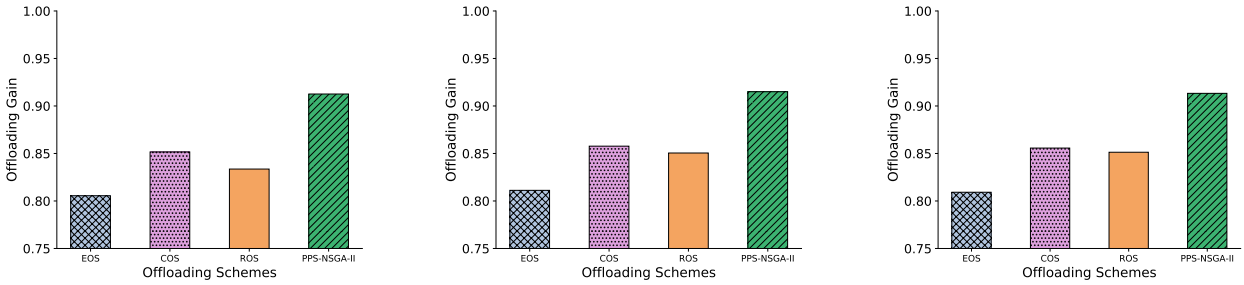
(a) $N = 10$ (b) $N = 50$ (c) $N = 100$

Fig. 4: Offloading gain of different offloading schemes for $w = 0.2$.



(a) $N = 10$ (b) $N = 50$ (c) $N = 100$

Fig. 5: Offloading gain of different offloading schemes for $w = 0.5$.



(a) $N = 10$ (b) $N = 50$ (c) $N = 100$

Fig. 6: Offloading gain of different offloading schemes for $w = 0.8$.

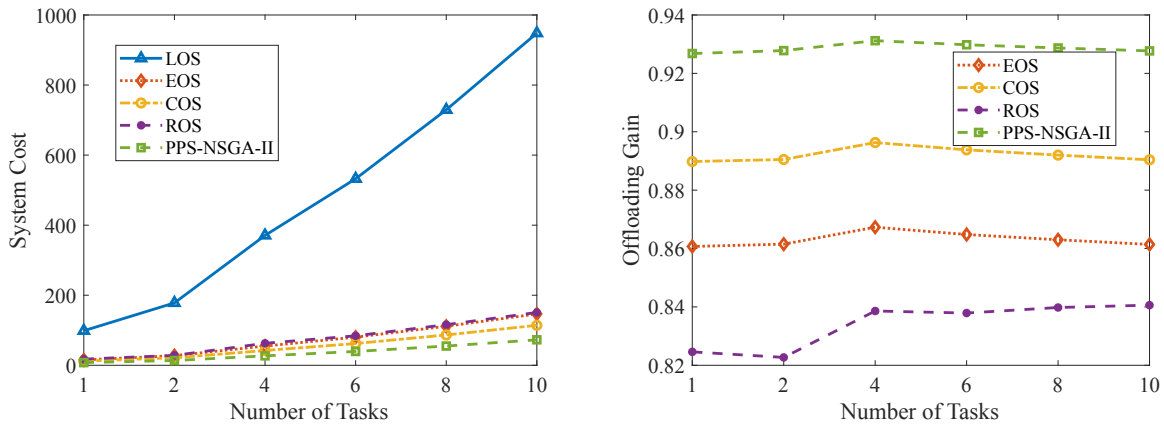


Fig. 7: System cost and offloading gain on different offloading schemes under different number of tasks.

EOS and COS may be negative when the wireless bandwidth is small, which means that a computing task should not be offloaded to edge or cloud servers due to large communication cost in the case wireless bandwidth is small enough.

Fig. 9 presents the performance of system cost and offloading gain on different offloading schemes under different edge server CPU frequency. The performance of LOS and COS do not change no matter what the CPU frequency of the edge server. With the increment of edge server CPU frequency, the performance of system cost and offloading gain of EOS grows faster than ROS and PPS-NSGA-II. However, PPS-NSGA-II still achieves the best results among all the offloading schemes.

E. Impacts of Different Types of Applications

Fig. 10 illustrates the performance of system cost and offloading gain on different offloading schemes under different types of applications. With the increment of parameter ρ of different types of applications, the computing delay increases directly. The system cost of LOS increases very fast due to the poor computing capability of MDs, while COS and PPS-NSGA-II grow slowly due to the powerful computing resources at the cloud servers. PPS-NSGA-II will make more offloading decisions to offload the tasks to cloud servers. On the other hand, the system cost of EOS and ROS grow gradually and the increasing speed of EOS is slower than ROS. Furthermore, the performance of offloading gain of EOS, COS and PPS-NSGA-II is much better than ROS, and the COS and PPS-NSGA-II achieve the best and similar results due to the increment of parameter ρ of different types of applications.

F. Comparison with Deep Learning Methods

Some deep learning-based methods are used to address the offloading problems, which often have two research directions. First, the labeled dataset of sample offloading decisions can be generated and then the labeled dataset will be used to train the deep neural networks [19], [20]. In addition, deep reinforcement learning-based methods are used to generate near-optimal offloading decisions [21], [22]. Compared with the proposed PPS-NSGA-II, the deep learning methods can obtain one solution in a single run based on system cost or offloading gain with a specific weight parameter w, while PPS-NSGA-II can achieve a set of non-dominated solutions. Furthermore, PPS-NSGA-II doesn't need the labeled database of offloading decisions. Single-objective optimization methods (such as GA and MA) also get single solutions once time. Other traditional methods such as branch and bound (BB) [45] and integer programming spend more time obtaining a single sub-optimal solution.

We compare PPS-NSGA-II with two deep learning-based methods DDLO [19] and DDRO [21] as well as BB [45] to solve computation offloading problems, where w is set to 0.5 as well as N is equal to 10. Fig. 11 shows the system cost and offloading gain obtained by DDLO, DROO, BB and PPS-NSGA-II. We can observe deep learning-based methods can get better results than the traditional BB method. However, PPS-NSGA-II still obtained the best offloading decision.

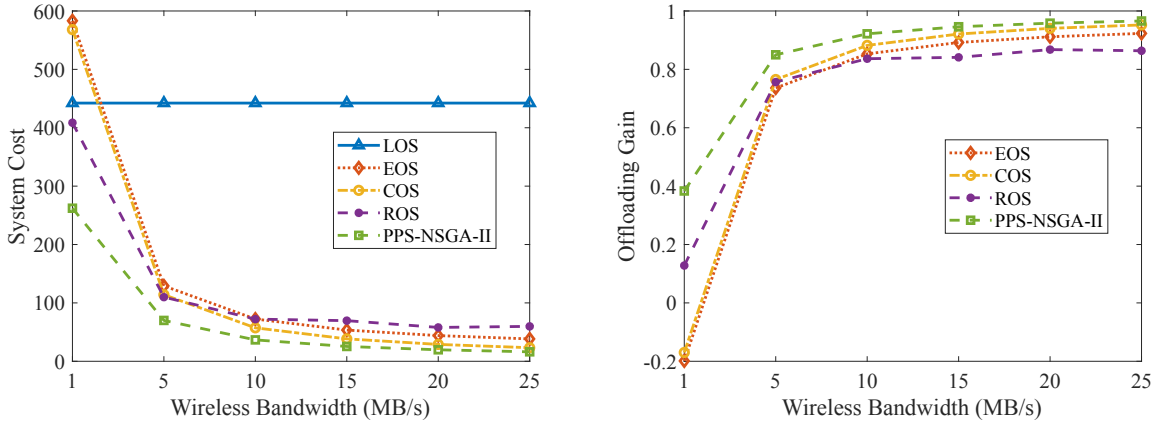


Fig. 8: System cost and offloading gain on different offloading schemes under different wireless bandwidth.

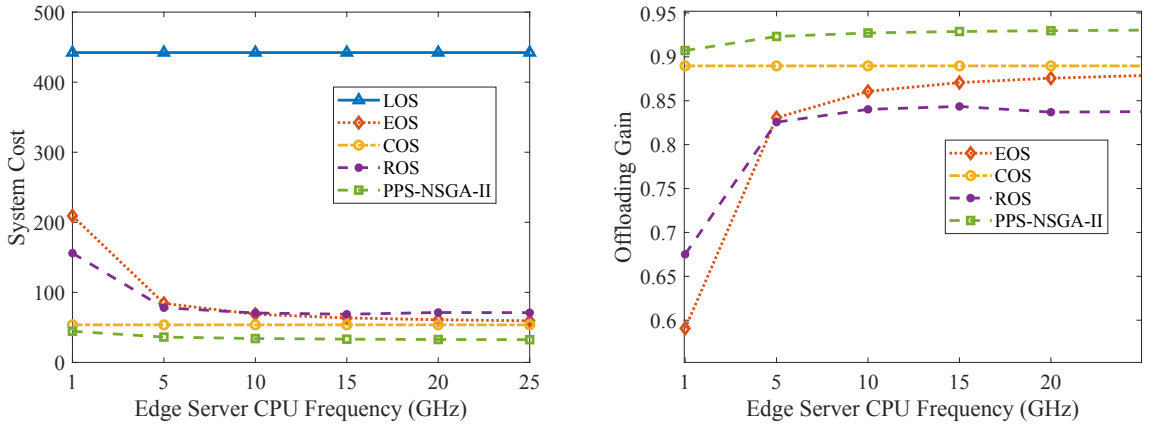


Fig. 9: System cost and offloading gain on different offloading schemes under different edge server CPU frequency.

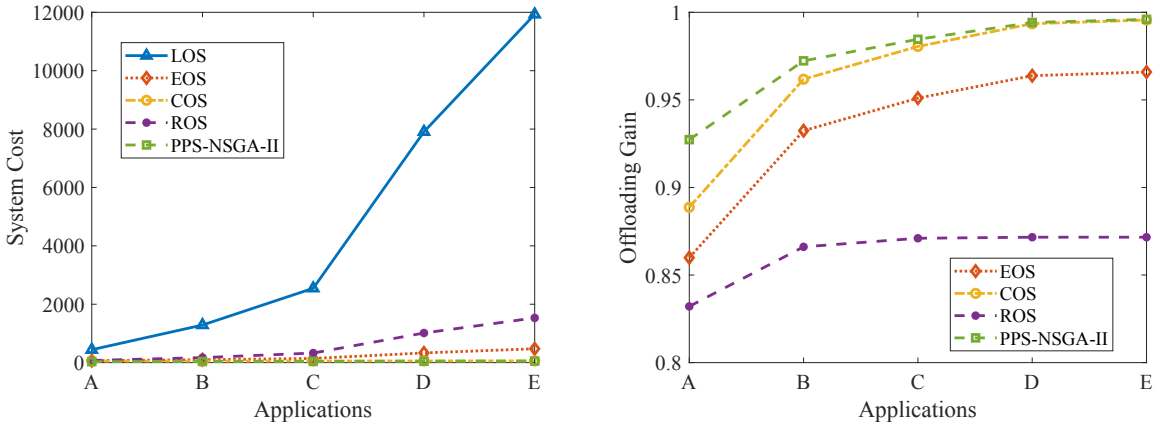


Fig. 10: System cost and offloading gain on different offloading schemes under different types of applications.

VI. Conclusion and Future Work

In this paper, three constrained multi-objective evolutionary algorithms (CMOEAs) are developed to solve IoT-enabled computation offloading problems in collaborative edge and cloud computing networks. We established a constrained multi-objective computation offloading model for minimizing time and energy consumption of IoT devices. NSGA-II, SPEA2 and SPEA2-SDE are embedded into PPS framework for solving CMOPs, and then PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE are realized. In the push search stage, PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE algorithms search for the unconstrained solutions without considering any constraints. In the pull search stage, the constraint handling principle (CDP) is integrated into these algorithms to pull the unconstrained solutions to approximate the constrained Pareto fronts. Three challenging constrained benchmark suites (LIR-CMOP, DAS-CMOP, and DOC) are used to test the performance of the proposed algorithms by comparing them to the other state-of-the-art CMOEAs. These algorithms are also adopted to solve the constrained multi-objective computation offloading problems, and compared the performance with other different offloading schemes. The experimental results

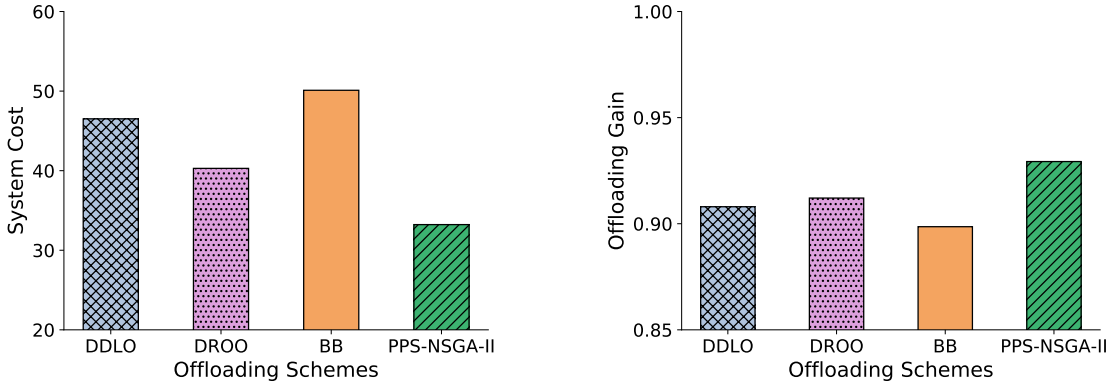


Fig. 11: System cost and offloading gain obtained by DDLO, DROO, BB and PPS-NSGA-II.

show the proposed algorithms can achieve better performance than other compared representative algorithms, and outperform other different offloading policies.

The different tasks in IoT devices are assumed to be independent in this work. In the future, the dependencies between the tasks in one application will be considered. In addition, other objectives in the computation offloading problems (e.g., monetary cost and security) in mobile edge-cloud computing networks will be investigated. Furthermore, the three approaches can be scale to mobility scenarios. The IoT devices move between different base stations during the offloading period that may influence the performance of the task offloading. Considering the mobility in the offloading process, there exists task migration and information handover between different base stations. Thus, we can set up a sub mobility delay model embedded in the offloading decision model. The response time constraint may change, and the three proposed algorithms can be used to solve the combined model to obtain solutions to satisfy the requirements.

REFERENCES

[1] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.

[2] L. P. Qian, Y. Wu, B. Ji, L. Huang, and D. H. Tsang, "Hybridiot: Integration of hierarchical multiple access and computation offloading for iot-based smart cities," *IEEE Network*, vol. 33, no. 2, pp. 6–13, 2019.

[3] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, p. 1446, 2019.

[4] F. Liu, Z. Huang, and L. Wang, "Energy-efficient collaborative task computation offloading in cloud-assisted edge computing for iot sensors," *Sensors*, vol. 19, no. 5, p. 1105, 2019.

[5] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

[6] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 200–10 232, 2020.

[7] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[8] M. S. Elbamby, C. Perfecto, C. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, "Wireless edge computing with latency and reliability guarantees," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1717–1737, 2019.

[9] Q. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.

[10] H. Wang, T. Liu, B. Kim, C. W. Lin, S. Shiraishi, J. Xie, and Z. Han, "Architectural design alternatives based on cloud/edge/fog computing for connected vehicles," *IEEE Communications Surveys Tutorials*, vol. 22, no. 4, pp. 2349–2377, 2020.

[11] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[12] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[13] S. Bi, L. Huang, and Y.-J. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, 2020.

[14] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An admm framework," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2763–2776, 2019.

[15] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[16] F. Wang, M. Zhang, X. Wang, X. Ma, and J. Liu, "Deep learning for edge computing applications: A state-of-the-art survey," *IEEE Access*, vol. 8, pp. 58 322–58 336, 2020.

[17] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[18] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[19] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile Networks and Applications*, pp. 1–8, 2018.

[20] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city internet of things," *IEEE Internet of Things Journal*, 2020.

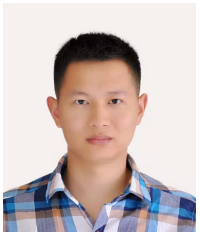
[21] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, 2019.

[22] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2020.

[23] L. Kuang, T. Gong, S. OuYang, H. Gao, and S. Deng, "Offloading decision methods for multiple users with structured tasks in edge computing for smart cities," *Future Generation Computer Systems*, vol. 105, pp. 717–729, 2020.

[24] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for iot-enabled cloud-edge

computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.
[25] M. Goudarzi, H. Wu, M. S. Palaniswami, and R. Buyya, "An application placement technique for concurrent iot applications in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, 2020.
[26] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, "Constraint handling in multiobjective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 514–525, 2009.
[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.
[28] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. You, J. Mo, C. Wei, and E. Goodman, "An improved epsilon constraint-handling method in moea/d for cmops with large infeasible regions," *Soft Computing*, vol. 23, no. 23, pp. 12 491–12 510, 2019.
[29] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.
[30] K. Li, R. Chen, G. Fu, and X. Yao, "Two-archive evolutionary algorithm for constrained multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 303–315, 2018.
[31] Z. Fan, W. Li, X. Cai, H. Li, C. Wei, Q. Zhang, K. Deb, and E. Goodman, "Push and pull search for solving constrained multi-objective optimization problems," *Swarm and evolutionary computation*, vol. 44, pp. 665–679, 2019.
[32] Y. Tian, T. Zhang, J. Xiao, X. Zhang, and Y. Jin, "A coevolutionary framework for constrained multi-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, 2020.
[33] Q. Zhu, Q. Zhang, and Q. Lin, "A constrained multiobjective evolutionary algorithm with detect-and-escape strategy," *IEEE Transactions on Evolutionary Computation*, 2020.
[34] Z. Fan, Z. Wang, W. Li, Y. Yuan, Y. You, Z. Yang, F. Sun, and J. Ruan, "Push and pull search embedded in an m2m framework for solving constrained multi-objective optimization problems," *Swarm and Evolutionary Computation*, vol. 54, p. 100651, 2020.
[35] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *IEEE transactions on evolutionary computation*, vol. 13, no. 2, pp. 284–302, 2008.
[36] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.
[37] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.
[38] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for pareto-based algorithms in many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 348–365, 2013.
[39] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing." *HotCloud*, vol. 10, no. 4-4, p. 19, 2010.
[40] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6353–6367, 2018.
[41] Y. Zhou, M. Zhu, J. Wang, Z. Zhang, Y. Xiang, and J. Zhang, "Tri-goal evolution framework for constrained many-objective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.
[42] Z.-Z. Liu and Y. Wang, "Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 870–884, 2019.
[43] Y. Tian, Y. Zhang, Y. Su, X. Zhang, K. C. Tan, and Y. Jin, "Balancing objective optimization and constraint satisfaction in constrained evolutionary multi-objective optimization," *IEEE Transactions on Cybernetics*, 2020.
[44] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE transactions on evolutionary computation*, vol. 10, no. 1, pp. 29–38, 2006.
[45] I. A. Elgendy, W. Zhang, Y.-C. Tian, and K. Li, "Resource allocation and computation offloading with data security for mobile edge computing," *Future Generation Computer Systems*, vol. 100, pp. 531–541, 2019.



Guang Peng received the B.E. and M.S. degrees from Xi'an Jiaotong University, China, in 2014 and 2016, respectively, both in electrical engineering. He is currently working towards the PhD degree at Freie Universität Berlin, Germany. His research interests include intelligent computation, multi-objective optimization, mobile cloud/edge computing and deep learning.

Huaming Wu received the B.E. and M.S. degrees from Harbin Institute of Technology, China, in 2009 and 2011, respectively, both in electrical engineering. He received the Ph.D. degree with the highest honor in computer science at Freie Universität Berlin, Germany in 2015. He is currently an associate professor in the Center for Applied Mathematics, Tianjin University. His research interests include model-based evaluation, wireless and mobile network systems, mobile cloud computing and deep learning.

Han Wu received his B.S. degree from Huazhong University of Science and Technology, Wuhan, in 2013 and M.S degree from Tongji University, Shanghai, in 2016. Supported by China Scholarship Council, he joined in Dependable Distributed Systems group of Freie Universität Berlin, supervised by Prof. Dr. Katinka Wolter since 2016. His research interests include performance evaluation, streaming processing, Docker technology and machine learning.

Katinka Wolter received her PhD degree from Technische Universität Berlin in 1999. She has been Assistant professor at Humboldt-University Berlin and lecturer at Newcastle University before joining Freie Universität Berlin as a professor for dependable systems in 2012. Her research interests are model-based evaluation and improvement of dependability, security and performance of distributed systems and networks.