Joint Optimization of Energy Consumption and Latency in Mobile Edge Computing for Internet of Things

Laizhong Cui, Chong Xu, Shu Yang, Joshua Zhexue Huang, Jianqiang Li, Xizhao Wang, Zhong Ming, Nan Lu

Abstract—**With wide adoption of Internet of Things across the world, the IoT devices are facing more and more intensive computation task nowadays. However, the IoT devices are usually limited by their computing capability and battery lifetime. Mobile Edge Computing provides new opportunities for developments of IoT, since edge computing servers which are close to devices can provide more powerful computing resources. The IoT devices can offload the intensive computing tasks to edge computing servers, while saving their own computing resources and reducing energy consumption. However, the benefits come at the cost of higher latency, mainly due to additional transmission time, and it may be unacceptable for many IoT applications. In this paper, we try to find a trade-off between energy consumption and latency, in order to satisfy user demands of various IoT applications. We formalize the problem into a constrained multi-objective optimization problem and find the optimal solutions by a modified fast and elitist nondominated sorting genetic algorithm (NSGA-II). To improve performance of the algorithm, we propose a novel problem-specific encoding scheme and genetic operators in the proposed modified NSGA-II. We also conduct extensive simulation experiments to evaluate the proposed algorithm and its sensitivity under certain major parameters. The experimental results show that the proposed algorithm can find a large number of optimal solutions to adjust the corresponding offloading decision according to the real-world situation.**

Index Terms—**Computation offloading, Constrained Multi-Objective Optimization (CMOP), Mobile Edge Computing (MEC), Internet of Things (IoT).**

I. Introduction

WITH the explosively increasing number of smart mobile devices (SMDs), e.g. smart phones, tablet personal computer, PDA smart terminal , wearable devices and virtual reality devices, the Internet of Things scales dramatically. The explosion of SMDs and the emergence of 5G networks have also brought out various new requirements to IoT applications [1]. More and more applications put forward higher demand for security, real-time and intelligence. Especially, in order to improve the Quality of Experience (QoE) of SMDs, the response time requirements for application requests are growing rapidly. However, SMDs are usually limited by their battery lifetime and computing capability, the contradiction between the demand for running complexity applications and the limited capability of SMDs becomes sharper gradually [2].

Due to the ever-changing user demands for variety of mobile applications, the QoE requirements for different SMD traffics are totally different. Some SMDs will generate computation intensive tasks, while others may generate latency-critical tasks. Although out-sourcing the tasks to traditional centralized cloud computing server can greatly save computing resources and reduce energy consumption in need [3], but it cannot satisfy the strict latency requirements. Especially when the traffic has to travel across the core network, the latency is much higher and non-deterministic. Thus, traditional cloud computing cannot get the best of both worlds, i.e. either the SMDs must consume lots of computation and battery resources to handle the intensive computing tasks, or the applications have to tolerate high latency.

Recently, Mobile Edge Computing (MEC) is seen as a promising technique to alleviate the problem mentioned above. MEC deploys servers with more powerful computing capability at the edge of networks, so that the latency can be greatly reduced by offloading tasks to the edge servers instead of the cloud. In this way, the intensive computing tasks can be immigrated from edge devices to MEC servers, thereby the computation resources and energy consumption can be saved, leading to a much longer lifespan of the entire IoT [4-5]. Many IoT scenarios can improve performance by deploying MEC server, e.g. traffic management, ocean monitoring, edge content caching and local content distribution [6].

Although MEC system can reduce latency compared with centralized cloud computing, the latency between SMDs and MEC servers is also not a neglectable variable, and it should be taken into consideration while assigning tasks among them. Taking 6 degree-of-freedom (DoF) virtual reality video for an example, the application delay required is generally less than 20ms [7]. However, the latency time and the quality of wireless communication are closely related. If there is more data transmitted in the channel, the latency time will be prolonged, so that the latency requirement of the application cannot be met.

This work was supported in part by the National Natural Science Foundation of China under Grants 61772345, 61732011, 61672358 and 61836005, Major Fundamental Research Project in the Science and Technology Plan of Shenzhen under Grants JCYJ20160310095523765 and JCYJ20160307111232895, Guangdong Pre-national project under Grant 2014GKXM054.

The authors are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, Guangdong, China

E-mail: {cuilz, yang.shu, zx.huang, lijq, xzwang, mingz, lunan} @szu.edu.cn, xuchong2016@email.szu.edu.cn

Shu Yang is the corresponding author.

In other words, latency and energy consumption issues also exist in the MEC system. In this paper, we investigate into the computation offloading problem and achieve a trade-off between energy consumption and latency by scheduling offloading tasks between SMDs and MEC servers.

In this paper, we focus on the computation offloading problem to find out the optimal trade-off between energy consumption and latency. Offloading all the tasks to MEC servers and computing locally are both extreme cases in the decision space. Offloading can reduce the computation and battery resources devoted SMDs, while computing locally can reduce transmission latency. Both mechanisms improve in one aspect, while fails in another, however, most IoT applications are constrained by both [8-10]. What is worse, the user demands may change over time, e.g. some medical wearable devices require less latency when detecting emergency events, and require longer battery lifetime if no event is detected. Thus, we need to setup a decision module that can output different decisions based on the given constraints.

Above all, we formalize the computation offloading problem into a constrained multi-objective optimization problem and find out the Pareto optimal set through the problem-specific NSGA-II algorithm. The main contributions of this paper are summarized as follows:

1) We use the mechanism of small cells [11], which is considered as a promising solution to improve spectrum efficiency and energy efficiency. The centralized MEC system with multi-cell in the IoT is considered in this paper. We use queue theory and optimization algorithm to model the local execution process, transmission process and MEC server execution process.
2) We model the computation offloading problem as a constrained multi-objective optimization problem which involves minimizing the average energy consumption of SMDs and the average latency time of tasks. The Pareto optimal set of this multi-objective optimization problem is obtained by the modified NSGA-II algorithm. There are various optimal solutions in the Pareto optimal set that can meet different actual requirements.
3) The effectiveness of the modified NSGA-II algorithm is evaluated by extensive simulations and the sensitivity of the algorithm under multiple parameters is analyzed.

The rest of this paper is organized as follows. The recent related works of computation offloading problem in MEC system are described in Section II. The system overview and formulated optimization problem are presented in Section III. Original version of NSGA-II [12] and the design of modified NSGA-II are proposed in Section IV. In Section V, simulation experiments are conducted, and results analysis are discussed. Finally, we conclude this paper in Section VI.

II. RELATED WORK

Computation offloading in MEC system is a challenging issue that has attracted many research works on it. In this section, we will extensively introduce recent research works and progresses on this topic. In [13], Mach *et al.* gave a detailed introduction to the MEC system and proposed two most commonly used indicators for evaluating performance, including energy consumption and latency time.

In [14], Liu *et al.* took the queue state of task buffer, the local CPU status and communication unit status into account. By analyzing the average latency of each task and the average energy consumption of SMDs, the authors considered minimizing the latency under power-constrained as the optimization objective and a one-dimensional search strategy was proposed to find out the optimal task scheduling policy. In [15], Wang *et al.* modeled two optimization problems and solved them separately. One was energy consumption of SMD minimization (ECM), the other was latency of application execution minimization (LM). In order to solve the ECM problem, the authors redefined it as a convex optimization problem and obtained its optimal solution. To handle the LM problem, a locally optimal algorithm with the univariate search technique was proposed. The authors in [16] aimed to minimize both total tasks' execution latency and the SMD's energy consumption by jointly optimizing the task offloading decision and the SMD's CPU-cycle frequency. Different methods were proposed to deal with two cases separately. Simulation results show that the (semidefinite relaxation) SDR-based algorithms achieve near optimal performance whether the CPU frequency is fixed or not. In [17], Wang *et al.* proposed a framework that can implement offloading decision and interference management. Three optimization problems were solved in this framework. The offloading decision was made by the MEC server based on the overhead estimation of SMDs and MEC server. Physical resource block (PRB) allocation was solved by the graph coloring method. The allocation of computing resources was based on the previous two results. In [18], Deng *et al.* proposed an adaptive sequential offloading game approach, and the approach can adaptively adjust the number of users in the offloading queue to avoid additional queuing delays when the network scale was expanded. In [19], Zhang *et al.* presented an integrated framework for computation offloading and resource allocation in MEC networks in order to investigating the trade-off between energy consumption and execution latency. And an energy-aware offloading scheme was proposed to minimize the weighted sum of energy consumption and latency time by optimizing computation offloading decision, local computation frequency scheduling, power allocation and channel allocation. The weighting factors of energy consumption and execution latency were associated with residual power of SMDs.

In addition, there are other indicators used as the optimization objectives in previous studies. In [20], Mao *et al.* proposed the execution cost of execution delay and task failure as performance indicators, and converted the problem into the execution cost minimization problem. The authors proved that the execution cost minimization problem is a high-dimensional Markov decision problem, and then proposed Lyapunov optimization-based dynamic computation offloading (LODCO) algorithm to solve the execution cost minimization problem. The proposed LODCO algorithm optimizes the offloading decision, the CPU-cycle frequencies and the transmit power of

SMD simultaneously. Simulations showed that the LODOC algorithm can not only reduce the execution cost, but also reduce the failure rate of the task. In [21], Bi *et al.* studied the optimal design in a multi-user wireless powered MEC network using binary computation offloading policy. They jointly optimized the offloading decision and the system transmission time allocation to maximize the weighted sum computation rate. Due to the strong coupling between computation offloading and time allocation in a multi-user mode, the common method is no longer effective. They solved this problem in three steps. First, a decoupled optimization method is proposed. Then, the optimal time allocation is obtained by a simple bi-section search algorithm. Finally, a coordinate descent method is devised to optimize the mode selection. In [22], Liu *et al.* investigated the E&D&P problem that jointly minimizes the energy consumption, latency time and payment cost in a fog computing heterogeneous network. A multi-objective optimization problem was formulated and the corresponding decision variables include offloading probability and transmit power of SMDs. They transformed the multi-objective optimization problem into a single-objective optimization problem by the scalarization method and Interior point method (IPM) was applied to handle it. In [23-24], in order to maximize a weighted profit for network operators, Huang *et al.* tried to increasing the total service traffic of the local cloud, and proposed an adaptive provisioning service update strategy while considering the access delay and virtual machine latency. The authors formulated the problem to be an integer linear programming and introduced a heuristic framework. The proposed strategy can provide operators with solutions about when and how to update the provisioning solution.

Although there are many indicators in the MEC system as mentioned in previous literature, we use the two most important indicators as the evaluation metrics in the paper, i.e., latency and energy consumption. The requirements for the response time make latency be a factor that was taken into consideration by most current works. On the other hand, sustainable development has become a research hotspot because of environmental degradation. In [25-26], the relationship between green challenges and big data has been studied. In [27], Wu *et al.* interpreted the sustainable development goals from the perspective of the information and communications technologies. Especially, in the field of Internet of Things, the issue of energy consumption has become more and more important for extending life span of the IoT. Furthermore, Wang *et al.* proposed an energy-efficient architecture for Industrial IoT (IIoT) [28].

In this paper, we jointly optimize energy consumption and latency, but we formulate the computation offloading problem as a constrained multi-objective optimization problem. The biggest difference from previous studies is that we obtain multiple optimal solutions instead of only one optimal solution. These solutions give the IoT applications based on MEC the power to choose offloading decisions according to the actual situations. The algorithm used in this paper is the modified NSGA-II algorithm which is a population-based heuristic algorithm.

III. SYSTEM OVERVIEW AND PROBLEM FORMULATION

In this section, we take an overview of the MEC system. We describe the network model, communication model, local computing model and edge computing model, respectively. Finally, we will formulate the offloading decision process into a constrained multi-objective optimization problem.

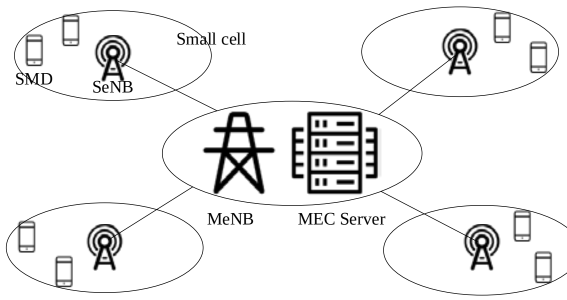


Fig. 1. Network model of the MEC system

A. System Overview

As shown in Fig. 1, we assume that there are many small cells in the IoT. All small cells are connected in a wired form to the Macro eNodeB (MeNB) where a MEC server is deployed [29]. The computing capability of MEC server is *F*. There are several Smart Mobile Devices (SMDs) and one Small cell eNodeB (SeNB) in each small cell. The communication mode between the SMDs and the corresponding SeNB is in a wireless manner. Each SMD is only connected to the SeNB in the same small cell. Let *NS* be the number of small cells. We enumerate the set of small cell by $m = \{1, 2, ..., NS\}$ and the SeNB in the *m*th small cell is referred to as S_m. Let U_m be the number of SMDs in the *m*th small cell, and $U_{m,j}$ be the *j*th ($j \in \{1, 2, ... U_m\}$) SMD in the *m*th small cell. Each SMD is modeled as a 3-tuple $U_{m,j} = (f_{m,j}, p_{m,j}, e_{m,j})$, where $f_{m,j}$ is the computing capability of $U_{m,j}$ and $p_{m,j}$ is transmission power when offloading tasks, $e_{m,j}$ is the power consumption coefficient when executing tasks locally.

We assume that SMD $U_{m,j}$ needs to handle several computation-intensive and latency-critical tasks which follow a Poisson process with an average arrival rate of $\lambda_{m,j}$ [30]. These tasks are independent with each other and they cannot be subdivided. Suppose that the amount of tasks generated by SMD $U_{m,j}$ is $K_{m,j}$. We model each task as a 2-tuple $\tau_{m,j,k} = \{d_{m,j,k}, c_{m,j,k}\}$, where $d_{m,j,k}$ is the data size of task $\tau_{m,j,k}$, and $c_{m,j,k}$ is the number of CPU cycles required to perform task $\tau_{m,j,k}$. Each task can be executed locally or offloaded to the MEC server to execute through the connected SeNB. The offloading decision is referred to as $O_{m,j,k}$. If $O_{m,j,k} = 1$, the task will be offloaded to MEC server. If $O_{m,j,k} = 0$, the task will be executed locally.

Throughout the computation offloading process, the energy consumption of SMD is divided into two components: energy consumption in performing tasks locally and energy consumption for offloading tasks. The latency of tasks consists of three parts: local execution latency, transmission latency and MEC execution latency. Based on this analysis, we model local computing, communication process and MEC server computing, respectively. The main notations in our system model are listed in Table I.

TABLE I
SUMMARY OF KEY NOTATIONS

Natation	Description
m	the set of small cells
S_m	the SeNB in the *m*th small cell
F	the computation capability of MEC server
U_m	the number of SMD in the *m*th small cell
$U_{m,j}$	the *j*th ($j \in \{1,2,...U_m\}$) SMD in the *m*th small cell
$f_{m,j}$	the computation capability of $U_{m,j}$
$p_{m,j}$	the transmission power of $U_{m,j}$ when offloading task
$e_{m,j}$	the power consumption coefficient when executing task locally
$\lambda_{m,j}$	average arrival rate of tasks for $U_{m,j}$
$K_{m,j}$	the number of tasks generated by SMD $U_{m,j}$
$\tau_{m,j,k}$	the *k*th task generated by $U_{m,j}$
$d_{m,j,k}$	the data size of task $\tau_{m,j,k}$
$c_{m,j,k}$	the number of CPU cycles required to perform $\tau_{m,j,k}$
$O_{m,j,k}$	offload decision of task $\tau_{m,j,k}$
B	the total bandwidth
N	the number of channels
$R_{m,j}$	the data transmission rate of $U_{m,j}$
$G_{m,j}$	the channel gain between SMD $U_{m,j}$ and SeNB S_m
σ^2	the background noise power
$I_{m,j}$	the interference at the $U_{m,j}$
$D_{m,j}$	the total data size which need to be offloaded
A	the total number of SMDs in the system

B. Communication Model

We assume that the total bandwidth is *B*, and there are *N* channels where the bandwidth of each is equal. The spectrum of small cells is reusable, and the communication mode between SMDs and SeNB in each small cell is Orthogonal Frequency-Division Multiple Access (OFDMA). Channel allocation is random and every SMD will only be assigned to one channel. Thus, the communication may only interfere with other SMDs which use the same channel in other small cells. In this paper, we only consider the task uploading from SMDs to SeNBs. Because the SeNBs and the MeNB communicate with each other through wired connection, on which the transmission time is almost negligible [18, 29]. The data size of task execution result returned from SeNB is usually very small. Thus, the time for transmission is much less than the time for uploading process, and we also ignore it in this paper [29, 31].

For the SMD $U_{m,j}$, if there exist tasks that need to be offloaded to the MEC server through SeNB, the uploading transmission rate $R_{m,j}$ can be calculated according to the Shannon-Hartley theorem as follows,

$$R_{m,j} = \omega \log_2 \left(1 + \frac{p_{m,j} G_{m,j}}{\sigma^2 + I_{m,j}} \right) \tag{1}$$

where $I_{m,j}$ is the interference parameter suffering from SMDs in other small cells on the same channel, and it can be obtained by the formula as follows,

$$I_{m,j} = \sum_{l=1, l \neq m}^{NS} \sum_{i=1}^{U_l} a_{l,i,m,j} p_{l,i} G_{m,l,i} \tag{2}$$

where $a_{l,i,m,j} \in \{0,1\}$ is a binary variable. If the channel used by $U_{m,j}$ and $U_{l,i}$ is the same, $a_{l,i,m,j} = 1$. Otherwise, $a_{l,i,m,j} = 0$. $p_{l,i}$ is the transmission power of $U_{l,i}$, and $G_{m,l,i}$ is the channel gain between SMD $U_{l,i}$ and SeNB S_m .

The time required to upload tasks is $t_{m,j}^{tran}$, it can be computed as follows,

$$t_{m,j}^{tran} = \frac{D_{m,j}}{R_{m,j}} = \frac{\sum_{k=1}^{K_{m,j}} O_{m,j,k} d_{m,j,k}}{R_{m,j}} \tag{3}$$

where $D_{m,j}$ is the total data size which need to be transmitted.

The energy consumption for the transmission process is the product of transmission power and transmission time, given as follows [32],

$$e_{m,j}^{tran} = p_{m,j} t_{m,j}^{tran} = \frac{p_{m,j} \sum_{k=1}^{K_{m,j}} O_{m,j,k} d_{m,j,k}}{R_{m,j}} \tag{4}$$

C. Local computing model

We assume the traffic model at $U_{m,j}$ follows the *M/M/1* queue [33]. Suppose the percentage of tasks executed locally is $PR_{m,j}$, the average remain time $t_{m,j}^{local}$ including queue delay and execution delay for these tasks is defined as follows [33],

$$t_{m,j}^{local} = \frac{1}{f_{m,j} - PR_{m,j} \lambda_{m,j}} \tag{5}$$

The energy required to perform task locally is $e_{m,j}^{local}$ which can be denoted as follows [32],

$$e_{m,j}^{local} = e_{m,j} t_{m,j}^{local} = \frac{e_{m,j}}{f_{m,j} - PR_{m,j} \lambda_{m,j}} \tag{6}$$

D. Edge computing model

The traffic model at MEC server also follows the *M/M/1* queue. For MEC server, the arrival rate of tasks can be computed as follows,

$$\lambda_{MEC} = \sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(1-PR_{m,j}\right)\lambda_{m,j} \tag{7}$$

The average remain time $t_{m,j}^{MEC}$ at MEC server is denoted as follows,

$$t_{m,j}^{MEC} = \frac{1}{F-\lambda_{MEC}} = \frac{1}{F-\sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(1-PR_{m,j}\right)\lambda_{m,j}} \tag{8}$$

E. Problem Formulation

In summary, the energy consumption of SMD that is the sum of local execution energy consumption and transmission energy consumption can be obtained from Eq. (4) and Eq. (6). The average energy consumption of SMD can be calculated as follows,

$$\begin{aligned} E &= \frac{1}{A}\sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(PR_{m,j}e_{m,j}^{local} + \left(1-PR_{m,j}\right)e_{m,j}^{tran}\right) \\ &= \frac{1}{A}\sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(\frac{PR_{m,j}e_{m,j}}{f_{m,j}-PR_{m,j}\lambda_{m,j}} + \frac{\left(1-PR_{m,j}\right)p_{m,j}\sum_{k=1}^{K_{m,j}}O_{m,j,k}d_{m,j,k}}{\omega\log_2\left(1+\frac{p_{m,j}G_{m,j}}{\sigma^2+I_{m,j}}\right)}\right) \end{aligned} \tag{9}$$

The latency time of tasks is the local execution latency or the sum of transmission latency and MEC execution latency. In this paper, we calculate the average latency of SMD and it can be obtained from Eq. (3), (5) and (8) as follows,

$$\begin{aligned} T &= \frac{1}{A}\sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(PR_{m,j}t_{m,j}^{local} + \left(1-PR_{m,j}\right)\left(t_{m,j}^{tran}+t_{m,j}^{MEC}\right)\right) \\ &= \frac{1}{A}\sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(\begin{aligned} &\frac{PR_{m,j}}{f_{m,j}-PR_{m,j}\lambda_{m,j}} + \frac{\left(1-PR_{m,j}\right)\sum_{k=1}^{K_{m,j}}O_{m,j,k}d_{m,j,k}}{\omega\log_2\left(1+\frac{p_{m,j}G_{m,j}}{\sigma^2+I_{m,j}}\right)} \\ &+\frac{\left(1-PR_{m,j}\right)}{F-\sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(1-PR_{m,j}\right)\lambda_{m,j}} \end{aligned}\right) \end{aligned} \tag{10}$$

where *A* is the total number of SMDs in the IoT.

At last, we formulate the offloading decision process as a multi-objective optimization problem which includes two objective functions. They are minimizing average energy consumption *E* in Eq. (9) and average latency time *T* in Eq. (10). The optimal problem **P1** can be formulated as follows.

$$\begin{aligned} &\mathrm{P1}: \min_{O_{m,j,k}}\{E,T\} \\ &s.\ t. \\ &C1: f_{m,j} > PR_{m,j}\lambda_{m,j} \\ &C2: F > \sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(1-PR_{m,j}\right)\lambda_{m,j} \\ &C3: O_{m,j,k} \in \{0,1\} \end{aligned} \tag{11}$$

The constraint C1 guarantees that the arrival rate of tasks that execute locally is less than SMD's service rate. C2 ensures that the arrival rate of tasks for MEC server is less than the MEC's service rate. C3 ensures that the offloading decision is binary.

IV. PROBLEM SOLUTION BASED ON MODIFIED NSGA-II

In general, there are two methods to solve multi-objective optimization problems. The first method is to transform the problem into a single-objective problem by giving each objective a different weight [34]. However, the weight value should be configured in advance, and the optimization result is greatly influenced by the configuration. When the users' demands change, the weight values need to be reset, and the algorithms need to be rerun. The second method is to use Multi-objective evolutionary algorithm (MOEA) [35-37] to obtain a set of optimal solutions which are called Pareto set. Then, the user can make further decisions based on real-world requirements. In this way, the system need not to re-run the algorithms even if the users' preferences change. In fact, NSGA-II algorithm is widely used for solving online problems [38-40]. Moreover, the network scale of the MEC systems is also smaller than that of the traditional networks. This feature also reduce the solution space of the problem, and thus the computation offloading problem can be suitable to be solved by NSGA-II algorithm. Therefore, we believe that NSGA-II algorithm can satisfy the time requirements for computation offloading problem. In this paper, we propose a modified NSGA-II and use this modified NSGA-II to solve the problem **P1**.

In this section, we will briefly introduce the original version of NSGA-II, and then describe the details of the modified NSGA-II and how to solve the problem **P1** by the modified NSGA-II.

A. The typical constrained multi-objective optimization problem

Without loss of generality, a constrained multi-objective optimization problems (CMOP) can be stated as follows,

$$\begin{aligned} &\text{Minimize} \quad F(x) = \left(f_1(x), f_2(x), \cdots, f_M(x)\right) \\ &\text{s. t.} \\ &\qquad g_i(x) \geq 0 \quad i = 1,\ldots P \\ &\qquad h_j(x) = 0 \quad j = 1,\ldots Q \\ &\qquad x \in \Omega \end{aligned} \tag{12}$$

where M is the number of objective functions. P is the number of inequality constraints, and Q is the number of equality constraints. Ω is the decision space. In the following paragraph, we present some background definitions related to MOP:

Definition 1. Pareto dominance

A solution x_1 dominates another solution x_2 (denoted by $x_1 \preceq x_2$) if and only if $f_k(x_1) \le f_k(x_2)$ for $\forall k \in \{1,\cdots,M\}$ and $\exists\, l \in \{1,\cdots,M\}$, satisfy $f_k(x_1) < f_k(x_2)$.

Definition 2. Pareto optimal set

For MOP $F(x)$, the Pareto optimal set is $P^* = \{x^* \in \Omega \mid \nexists\, x \in \Omega, x \preceq x^*\}$, where x^* is Pareto optimality.

Definition 3. Pareto optimal front

For MOP $F(x)$ and its Pareto optimal set P^*, the Pareto optimal front is $PF^* = \{F(x^*) \mid x^* \in P^*\}$.

The multi-objective optimization algorithm will find out a few optimal solutions since these multi-objective problems are conflicting each other. For the series of Pareto solutions obtained by the multi-objective optimization algorithm, there does not exist obvious difference between these solutions without extra conditions. The Pareto solutions are obtained through an iterative process. An allocation is defined as "Pareto optimal" when no further Pareto improvements can be made.

B. A fast and elitist nondominated sorting genetic algorithm: NSGA-II

NSGA-II that was proposed by Deb *et al.* [12] has become one of the most popular multi-objective evolution algorithms. Due to its simplicity and effectiveness, NSGA-II attracts many scholars to study it, and our work also draw lessons from these previous studies. The NSGA-II algorithm is showed in Algorithm 1. The population is initialized by random means. Then, the current generation population is sorted fast according to non-domination levels into fronts. The non-dominated sorting process is performed according to Definition 1. Each individual will be assigned to the corresponding front as shown in Fig. 2. Then, the crowding distance as shown in Fig. 3 is calculated for each individual on the basis of the objective functions to measure how close this individual is to its neighbors, which aims to effectively maintain diversity and spread of solutions. A binary tournament selection is adopted to construct the parent population, and the individuals in parent population are selected based on the rank of the assigned front and the crowding distance. An offspring population is generated by the operations of simulated binary crossover and polynomial mutation in the selected parent population. Thereafter, the new population that consists of the current generation population and the offspring population is sorted again according to the front rank and the crowding distance. Only the best individuals in the combined population are selected to guarantee elitism, and maintained to create the next generation. The algorithm repeats the above steps to precede the population's evolution until the maximum number of generations to decide termination is reached. In the following subsection, we will explain the steps of NSGA-II and the modified version in detail.

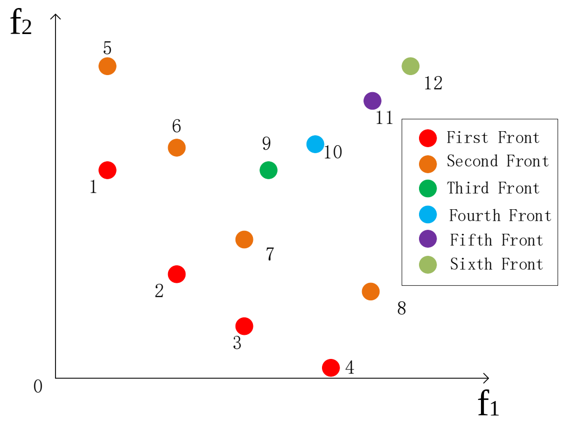


Fig. 2. Solution space of a problem with two objectives to be minimized

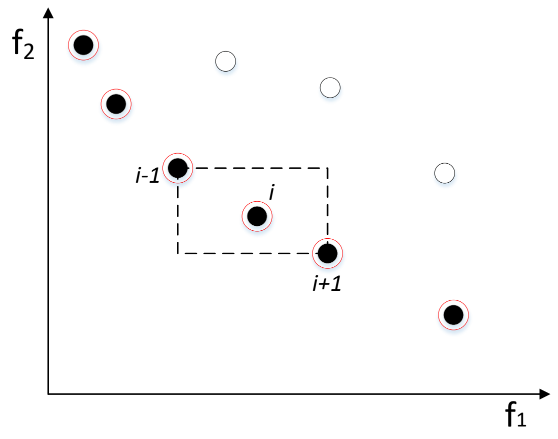


Fig. 3. Crowding distance calculation for the *i*th solution.

Algorithm 1: NSGA-II

```
1:   Initialize the parameters for NSGA-II.
2:   Generate an initial population by random means.
3:   Calculate multiple function value of each
     chromosome in the population.
4:   Rank the population by fast non-dominated sorting
     approach.
5:   Compute the crowding distance.
6:   Population are sorted based on their ranks and
     crowding distance.
7:   Produce new offspring by selection, crossover, and
     mutation operations.
8:   Calculate multiple function value of offspring.
9:   Combining the parent population and offspring
     population into new population.
10:  Rank the new population by fast non-dominated
     sorting approach.
11:  Compute the crowding distance of new population.
12:  Create a new parent population according to the sorted
     result.
13:  If the termination condition is not accomplished,
     return line 4.
14:  Return Pareto optimal set.
```

C. Offload decision through the modified NSGA-II

1) Initializing the parameters of NSGA-II

The main parameters of NSGA-II i.e., the population size *NP*, the maximum number of iterations *G*, the probability of crossover *CR* and the probability of mutation *PM* are initialized.

2) Generate an initial population

In this paper, each chromosome represents a set of offload decisions. Logically speaking, the value of each gene of chromosome must be 0 or 1, representing the offloading decision of its corresponding task. But in this paper, we design that each gene corresponding to one SMD. That is to say, the number of genes of each chromosome in the population is equal to the number of SMDs in the IoT. Suppose that SMD $U_{m,j}$ has $K_{m,j}$ tasks need to be decided, then the values for its associated gene $V \in \{0,1,2,\cdots,2^{K_{m,j}}-1\}$. Convert V to a binary representation, whose corresponding value is the offloading decision for all tasks of this SMD.

In addition, we generate the initial population in two steps. Assume that the number of genes is *A*. First, we generate a chromosome in which all genes are $G_x = 2^{K_x}-1 \ \left(x \in \{1,2,\cdots,A\}\right)$, and a chromosome in which all genes are 0. Then, we generate *A* chromosomes, each chromosome of these A chromosomes has one and only one gene is 0, and the rest genes are $2^{K_x}-1$. Finally, the rest $NP-A-2$ chromosomes are generated randomly. The initial population is shown in Fig. 4.

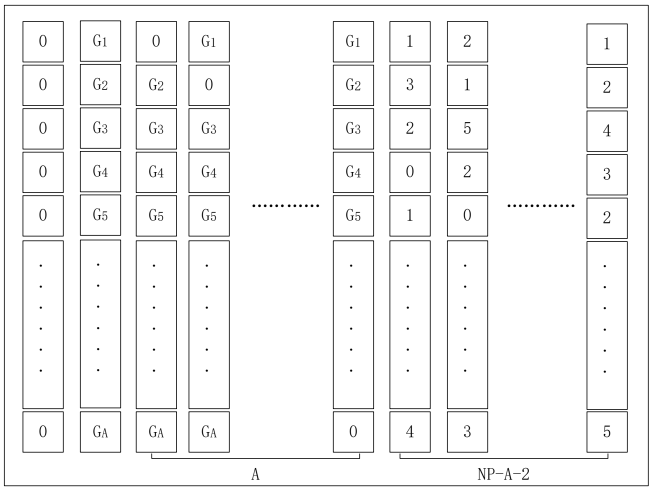


Fig. 4. Initial Population

3) Evaluating the solutions

After initializing the population, we evaluate each chromosome by Eq. (9) and (10). Note that there may be some chromosomes that do not satisfy the constraints in Eq. (11), we call these chromosomes infeasible solutions. Here, we define a variable called *CV* (constraint violation), and the value of *CV* can be calculated as follows.

$$CV_{m,j}^{1} = \begin{cases} 0, & if \ \ f_{m,j} > PR_{m,j}\lambda_{m,j} \\ \left(PR_{m,j}\lambda_{m,j} - f_{m,j}\right) / f_{m,j}, & otherwise \end{cases} \tag{13}$$

$$CV^{2} = \begin{cases} 0, & if \ \ F > \sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(1-PR_{m,j}\right)\lambda_{m,j} \\ \dfrac{\sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(1-PR_{m,j}\right)\lambda_{m,j} - F}{F}, & otherwise \end{cases} \tag{14}$$

$$CV = \sum_{m=1}^{NS}\sum_{j=1}^{U_m}\left(CV_{m,j}^{1}\right) + CV^{2} \tag{15}$$

The value of *CV* can reflect their distance from the feasible domain.

4) Fast non-dominated sorting approach

In this step, we stratify the population and all chromosomes will be planned into different fronts. The basis of stratification is the domination level of chromosomes. Since the problem **P1** is a constraint problem, we first redefine the concept of Pareto dominance (Definition 1).

Definition 4. Pareto constraint-dominance

A solution x_1 constraint-dominates another solution x_2 if any of the following conditions can be satisfied:

(1) x_1 is a feasible solution and x_2 is a feasible solution;

(2) x_1 and x_2 are feasible solutions and $x_1 \preceq x_2$;

(3) x_1 and x_2 are infeasible solutions and $CV(x_1) < CV(x_2)$.

We defined the nondominated solution that a solution is not dominated by any one solution in the population. First, we look for all non-dominated solutions in the entire population to form the first front. Then, we find the non-dominated solutions in the remaining solutions to form the second front. Repeat this process until all solutions are placed at the right front.

5) Crowding Distance Calculation

After stratifying the population, the density of solutions surrounding the specific solution is evaluated by a measure named crowding distance (*CD*). The calculation method of *CD* can be understood through Fig. 3 (in the case of two objective functions). For solution *i*, consider the two nearest solutions in the same front as the vertices to form rectangle. The *CD* of solution *i* is the average side length of this rectangle.

The specific calculation formula is given as follows,

$$CD_i = \begin{cases} \infty, & if \ \ i=1 \ \ or \ \ l \\ \dfrac{\sum_{k=1}^{M}\left(f_k(x_{i+1}) - f_k(x_{i-1})\right)}{f_k^{\max} - f_k^{\min}}, & otherwise \end{cases} \tag{16}$$

where *l* is the number of solutions in the same front, and *M* is the number of objection functions. $f_k^{\max}$ and $f_k^{\min}$ are the maximum and minimum values of the *k*th objection function.

6) *Selection operator*

For any two solutions x_1 and x_2, x_1 is better than x_2 if any of the following conditions is established:

1) The domination rank of x_1 is smaller than x_2;

2) x_1 and x_2 belong to the same front, but $CD_{x_1} > CD_{x_2}$.

According to this principle, all solutions will get a final sort. The better the sorting is, the higher the probability of being selected as the next generation of chromosome is.

7) *Modified Crossover and mutation operators*

In this step, we randomly choose two solutions in the population and generate a random value in range (0, 1). If this value is bigger than *CR*, the mutation operator will be done. Otherwise, the crossover operator will be performed.

As shown in Fig. 5, the mutation operation is performed as follows,

$$x'_{i,j} = \begin{cases} 2^{k_i} - 1 - x_{i,j}, & \text{if } rand < PM \text{ or } j = jrand \\ x_{i,j}, & \text{otherwise} \end{cases} \quad (17)$$

where x'_i is the offspring of x_i, $j \in [1, A]$ is the gene of chromosome, k_i is the number of tasks of *i*th SMD, $j = jrand$ makes sure at least one gene changes.

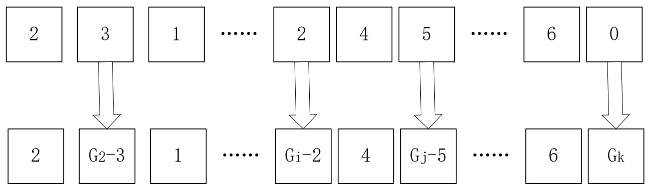


Fig. 5. Mutation Operation

As shown in Fig. 5, the crossover operation is performed as follows,

$$\begin{cases} x'_1 = round\left(\vartheta x_1 + (1-\vartheta) x_2\right) \\ x'_2 = round\left((1-\vartheta) x_1 + \vartheta x_2\right) \end{cases} \quad (18)$$

where both x'_1 and x'_2 are offspring, and ϑ is a random variable in range [0, 1] ($\vartheta = 0.2$ in Fig. 6). And *round*() guarantees that the value must be an integer.

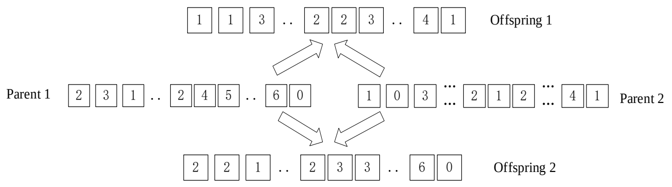


Fig. 6. Crossover Operation

8) *Generating new population*

In this step, the new population will be produced. After the generation of offspring population, we use the same method in step (3) to estimate their objective function values and constraint violation values. Then, the entire population including parent population and offspring are ranked based on Fast non-dominated sorting approach and the *CD* of each chromosome is calculated. Finally, the entire population are sorted based on their fronts and *CDs*. The best *NP* chromosomes are selected to form the next generation.

Finally, the algorithm judges whether the termination condition is met. If it is true, the first front is output as the optimal solutions, and converting these solutions into binary representation are the optimal decision. Otherwise, the algorithm returns the step (4), and the process loops until the termination condition is satisfied.

This main procedure of this algorithm is shown in Fig. 7.

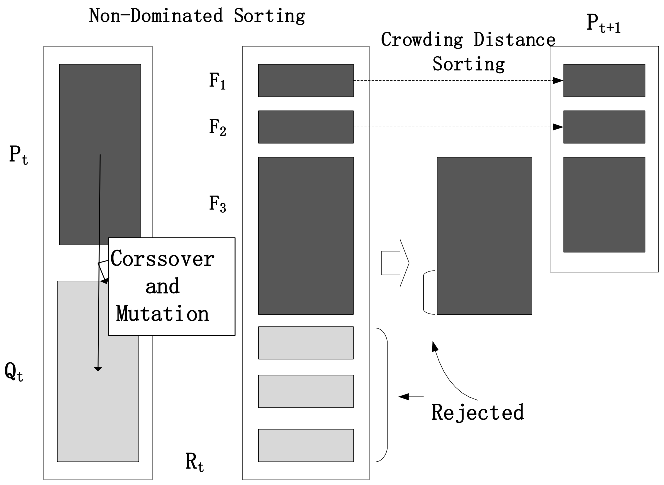


Fig. 7. NSGA-II procedure

V. SIMULATION AND RESULTS ANALYSIS

In order to verify the performance of the proposed modified NSGA-II in solving the computational offloading problem, we conduct extensive experiments. In addition, we analyze the sensitivity of this algorithm under several major parameters through experiments and verify the robustness of the proposed modified NSGA-II.

A. *Experimental Setup*

TABLE II
SIMULATION PARAMETERS

Parameter	**Value**
Number of SeNBs, *M*	5
Average arrival rate of tasks for $U_{m,j}$, $\lambda_{m,j}$	2
Computation capability of MEC server, F	30GHz
Computation capability of SMD $U_{m,j}$, $f_{m,j}$	[0.5, 1]GHz
Transmission power of $U_{m,j}$, $p_{m,j}$	300mW
Power consumption coefficient of $U_{m,j}$, $e_{m,j}$	16
Data size of task $\tau_{m,j,k}$, $d_{m,j,k}$	[500, 1000]Kb
Number of CPU cycles required to perform task $\tau_{m,j,k}$, $c_{m,j,k}$	[0.1, 0.5]GHz
Bandwidth *B*	20MHz
Number of channel *N*	10

We run the experiments on PCs with an Intel Core i5 CPU (3.2 GHz and 8 GB of RAM). The algorithms are implemented with Matlab R2017a. As shown in Fig. 1, the MEC network in this paper is centralized and the radius of network is 100m. The MeNB is at the center of the entire network, *M* SeNBs are evenly deployed in the network and *A* SMDs are deployed randomly in the network. The MEC server is placed on the MeNB. Every SMD is connected wirelessly to the nearest SeNB and wired communications are setup between SeNBs and MeNB. The configurations of the major parameters in the simulations are summarized in Table II. In order to study the sensitivity of each main parameter, we select the range of values of the corresponding parameters from [18-22], and use the intermediate value as the default value. The default values of the parameters are listed in Table II unless otherwise stated.

The parameters used in proposed modified NSGA-II are listed in Table III.

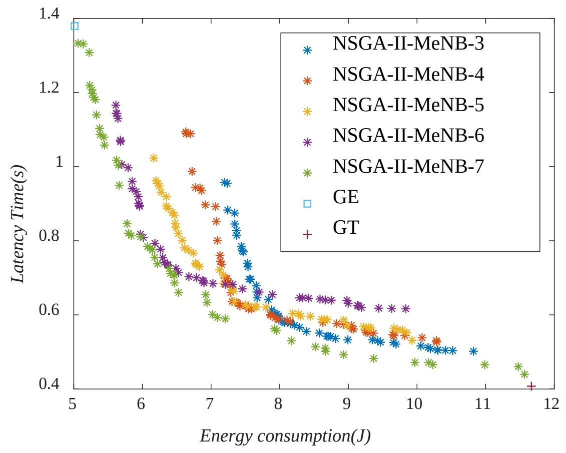
TABLE III
PARAMETERS OF THE PROPOSED MODIFIED NSGA-II

Parameter	**Value**
Population size *NP*	100
The maximum number of iterations *G*	100
Probability of crossover *CR*	0.8
Probability of mutation *PM*	0.3

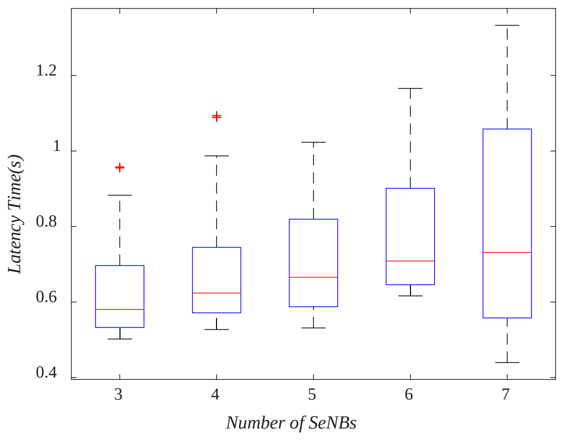
B. The impact of number of SMDs

In this subsection, we consider the impact of number of SMDs. In our network, the SMD is associated with SeNB and the number of channel *N* is limited, in order to ensure that there is no wireless communication interference in the same small cell. We adjust the number of SMDs by changing the number of MeNBs. The number of SMDs that each MeNB can provide services is 3-8. Other parameters are listed in Table II. The results are illustrated in Fig. 8.

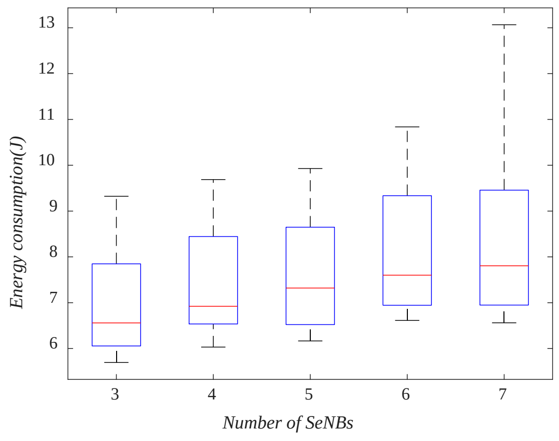
As shown in Fig. 8(a), we can observe that the proposed modified NSGA-II is able to get a balance between GE and GT. That is to say, the proposed modified NSGA-II can find more optimal decision-making options for the system. Fig. 8(b) and Fig. 8(c) show the impact of the number of SMDs on the overall network. We can observe that the range of Pareto



(a) Pareto front



(b) Box plots of the latency time in different number of SeNBs



(c) Box plots of the energy consumption in different number of SeNBs

Fig. 8. The impact of number of SMDs

optimal set is getting bigger as the number of SMDs increases, namely, the number of optimal solutions has increased. This is because the number of tasks that need to be assigned has increased with the size of the network, and the dimension of decision variables has also become larger.

In addition, both the average latency time and energy consumption increase with the number of SMDs. The reasons for the relationship between average latency time and number of SMDs are summarized as follows: 1) on the one hand, increasing the amount of tasks makes the execution time of the tasks in the MEC longer. 2) on the other hand, increasing the number of small cells makes the interference of wireless communication become more serious, resulting in slow communication speed and the communication time becomes longer.

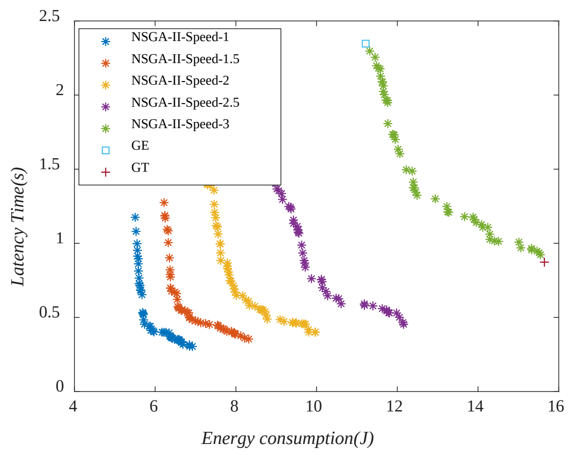
The main reason for the relationship between average energy consumption and number of SMDs is that the computation capability of the MEC server is also limited. As the number of tasks offloaded into the server reaches a certain number, more tasks are forced to be executed locally, and thus the average energy consumption of network becomes higher.

C. The impact of number of tasks

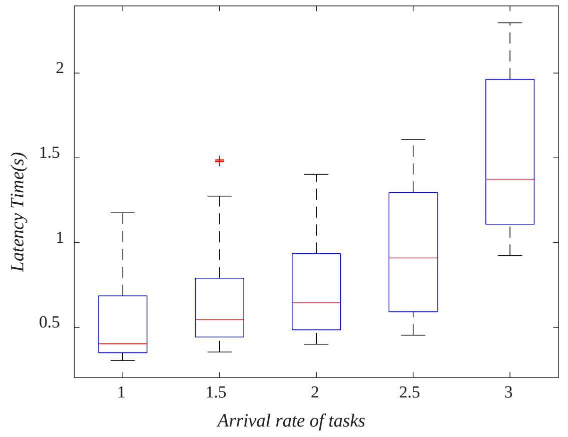
In this paper, the number of tasks is controlled by the arrival rate $\lambda_{m,j}$. We study the impact of task number on system performance by changing the value of $\lambda_{m,j}$. In the simulation, the value of $\lambda_{m,j}$ changes from 1 to 3 with the step size is 0.5. Fig. 9 shows the results of the simulation.

In Fig. 9(a), we can find that the proposed modified NSGA-II can find multiple Pareto optimal solutions between GT and GE. The impact of $\lambda_{m,j}$ on latency time and energy consumption can be observed in Fig. 9(b) and Fig. 9(c), respectively. The impact of number of tasks is the same as the impact of SMDs, and the number of optimal solutions increases with the number of tasks, as shown in Fig. 9(a). The reason for the relationship is the same with that illustrated in Section V-B. In Fig. 9(b) and Fig. 9(c), we can see that latency time and energy consumption significantly increase with $\lambda_{m,j}$, this is because the workload increases rapidly under this situation.

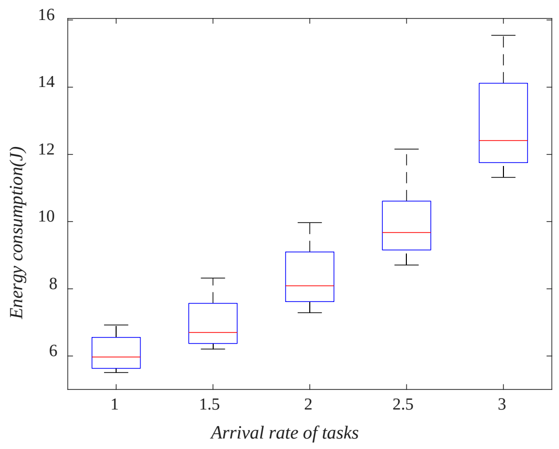
When the computation resources and communication resources of entire network keep the same, increasing number of tasks will inevitably lead to heavier burden for SMDs and MEC server. This result directly leads to a slower task processing and higher energy consumption. This phenomenon is not obvious when $\lambda_{m,j}$ is small, but it becomes worse as the arrival rate of tasks becomes faster and faster.



(a) Pareto front



(b) Box plots of the latency time in different arrival rate of tasks



(c) Box plots of the energy consumption in different arrival rate of tasks

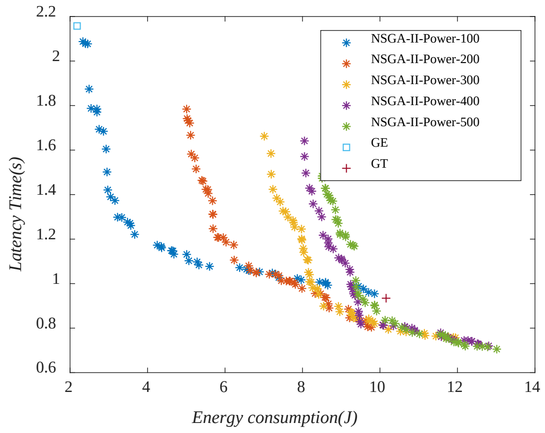
Fig. 9. The impact of number of tasks

D. The impact of transmission power

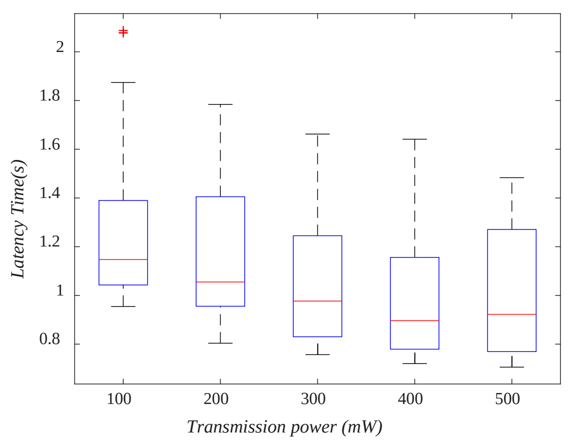
The relationship between the transmission power and the system performance is studied. In this experiment, the range of transmission power varies from 100 to 500 mW.

In Fig. 10(a), we can see that the proposed modified NSGA-II can get multiple Pareto optimal solutions between extreme values when transmission power is 100 mW. In fact, when the transmission power varies, the proposed modified NSGA-II can always find multiple Pareto solutions between extreme values. It is worth noting that the obtained Pareto solutions have a high degree of overlap when the transmission power is greater than 300. The results show that the impact on the offloading decision becomes very small when the transmission power reaches a certain level.

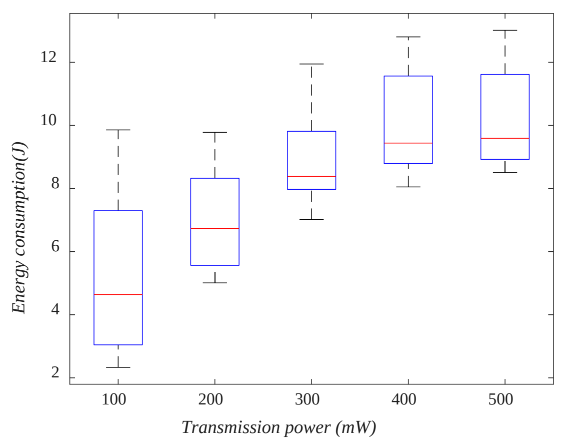
Logically speaking, the larger the transmission power is, the faster the uplink data speed is, and it will result in a smaller delay, which can be shown in Fig. 10(b). In contrast, the relationship between energy consumption and transmission power shown in Fig. 10(c) is positively correlated. In addition to the obvious relationship, we should also notice the slowdown in the trend. The degree of change for both latency time and energy consumption is shrinking gradually although the power is evenly increased. For a network with a fixed task volume, the effect of transmission power is not very large compared with other variables.



(a) Pareto front

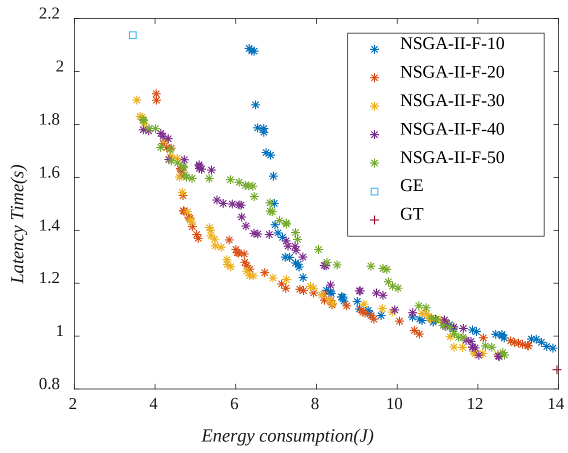


(b) Box plots of the latency time in different values of transmission power

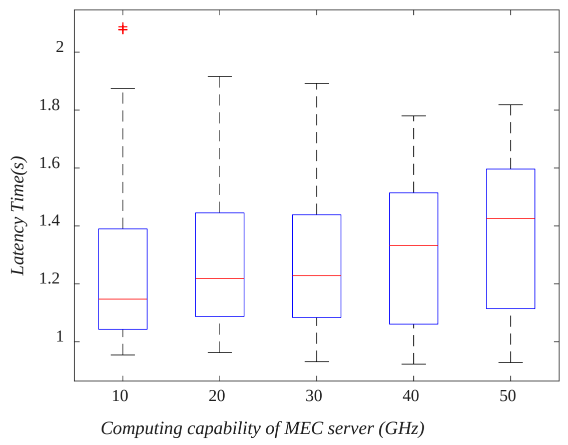


(c) Box plots of the energy consumption in different values of transmission power

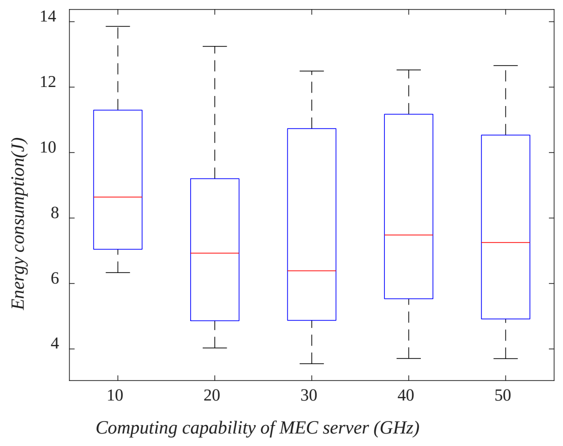
Fig. 10. The impact of the transmission power of SMDs



(a) Pareto front



(b) Box plots of the latency time in different computing capability of MEC server



(c) Box plots of the energy consumption in different computing capability of MEC server

Fig. 11. The impact of computing capability of MEC server

E. The impact of computing capability of MEC server

The experiments in this part are that how the objective functions change with the increasing computing capability of MEC server. The range of computing capability of MEC server is [10, 50].

It can be observed that the bound of the Pareto optimal solutions is similar with the increase of MEC server computing capability unless F is 10GHz. The increase in server computing capability allows it to calculate more tasks which increases the proportion of offloadable tasks, making the scope of the feasible area becomes larger. This is why the Pareto front is worse when F is 10GHz. But there is a threshold so that all tasks in the network can be offloaded to the MEC server without causing congestion. After that, the computing capability of the MEC server has little impact on the offload decision. This can be seen from Fig. 11(a), when F is bigger than 20GHz, the Pareto fronts is almost in the same location.

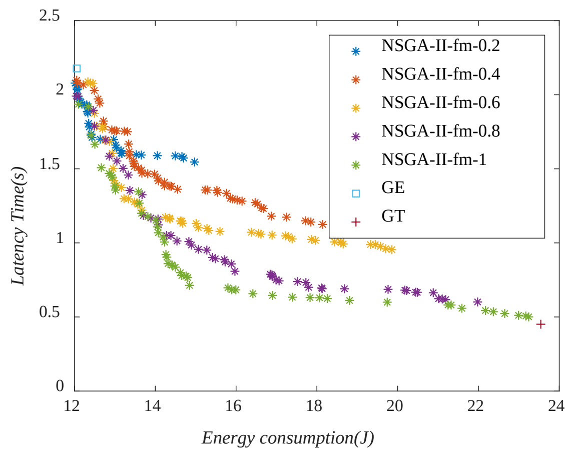
The same conclusion can be drawn from Fig. 11(b) and Fig. 11(c). There is no obvious difference between the distributions of the Pareto solution sets for different parameters.

The shortest latency time has hardly changed. And the energy consumption is reduced at the beginning, then remains unchanged. When the computing capability of the MEC server is sufficient for all computation tasks, the maximum offload proportion of tasks will no longer change as computing capability increases. This makes the lowest energy consumption and maximum latency time of the network will only change slightly.

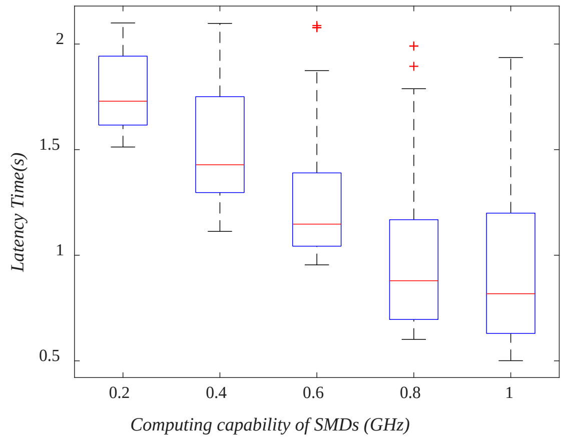
F. The impact of computing capability of SMDs

We focus on the influence of computing capability of SMDs on latency time and energy consumption in this subsection. The computing capability of SMDs is no longer randomly acquired, and all SMDs' computing capability is specified as the same value, including {0.2, 0.4, 0.6, 0.8, 1} GHz.

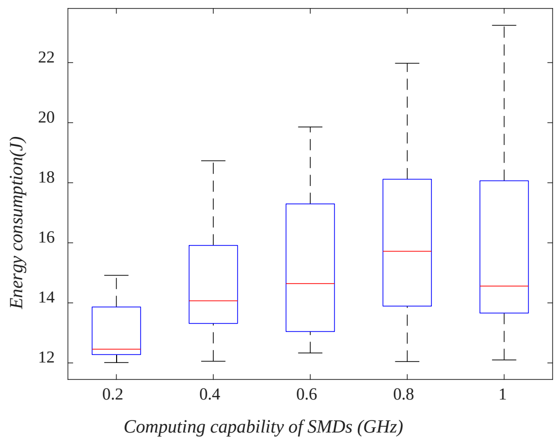
Fig. 12 shows the performance of network against the computing capability of SMDs. From Fig. 12(a), we can see that the density of Pareto optimal solutions with different parameters is different. But the extreme point in the upper left of Fig. 12(a) has the minimum energy consumption and the maximum latency time, and the value for each Pareto optimal set is almost the same. The lower computing capability of SMDs make the scope of the solutions smaller and denser. If the computing capability of SMDs is too small, it only can choose to offload its task to the MEC server for execution, resulting in a reduction in the feasible solution. In contrast, the number of feasible solutions increases.



(a) Pareto front



(b) Box plots of the latency time in different computing capability of SMDs



(c) Box plots of the energy consumption in different computing capability of SMDs

Fig. 12. The impact of computing capability of SMDs

We can also get the trend of latency time and energy consumption with computing capability of SMDs from Fig. 12(b) and Fig. 12(c). As the computing capability of SMDs increases, the possibility of tasks being executed locally becomes larger, which makes the latency time smaller and the energy consumption larger. It can be seen from the three figures that the computing capability of SMDs has a great impact on the network performance.

VI. Conclusion

In this paper, we investigated the problem of computation offload in MEC system for IoT. The centralized MEC network with multi-cells is considered in this paper. We built four models for calculating energy consumption and latency. In order to obtain a trade-off between average energy consumption of SMDs and latency of tasks, we formulate this problem as a constrained multi-objective optimization problem and solve it by the proposed modified NSGA-II algorithm. Simulation results showed that the proposed modified NSGA-II algorithm can always obtain the Pareto optimal set among the extremes which are computed by the greedy algorithms. In addition, we study the robustness of the proposed modified NSGA-II on the computation offloading problem by studying the performance of the proposed modified NSGA-II under different key parameters. Our research provides a better way than existing studies which can only find an optimal solution. The Pareto optimal set gives the system more flexible choices based on the requirements of different IoT applications. In the future, we will take more factors, such as allocating and scheduling of channels in the communication model on the decision-making process, into consideration.

References

[1] R. Atat, L. Liu, H. Chen, J. Wu, H. Li and Y. Yi, "Enabling cyber-physical communication in 5g cellular networks: challenges, spatial spectrum sensing, and cyber-security." *IET Cyber-Physical Systems: Theory & Applications*, vol. 2, no. 1, pp. 49-54, 2017.

[2] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu and B. Li, "Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications," *IEEE Wirel. Commun.*, vol. 20, no. 3, pp.14–22, Jun. 2013.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[4] W. Yu, F. Liang, X. He, Hatcher, W. G., C. Lu, J. Lin, and X. Yang. "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, no. 99, pp. 6900-6919, Mar. 2018.

[5] R. Wan, N. Xiong and N. T. Loc, "An energy-efficient sleep scheduling mechanism with similarity measure for wireless sensor networks," *Hum. Cent. Comput. Inf. Sci.*, vol. 8, no. 1, pp. 18, Jun. 2018.

[6] E. Ahmed and M. H. Rehmani, "Mobile Edge Computing: Opportunities, solutions, and challenges", *Future Gener. Comp. Sy.*, vol. 70, no. 2017, pp. 59-63, Sep. 2016.

[7] Qualcomm. "Augmented and Virtual Reality: The First Wave of 5G Killer Apps." [Online]. Available:https://www.qualcomm.com/documents/augmented-and-virtual-reality-first-wave-5g-killer-apps, Feb. 2017.

[8] V. Bhanumathi and C. P. Sangeetha, "A guide for the selection of routing protocols in WBAN for healthcare applications." *Hum. Cent. Comput. Inf. Sci.*, vol. 7, no. 24, Aug. 2017.

[9] B. Kim, "A Distributed Coexistence Mitigation Scheme for IoT-Based Smart Medical Systems." *J. Inf. Process Syst.*, vol. 13, no. 6, pp. 1602-1612, Dec. 2017.

[10] C. Kerang, H. Lee and H. Jung, "Task Management System According to Changes in the Situation Based on IoT." *J. Inf. Process Syst.* vol. 13, no. 6, pp. 1459-1466, Dec. 2017.

[11] S. Bu and F. R. Yu. "Green cognitive mobile networks with small cells for multimedia communications in the smart grid environment." *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2115-2126, Jun. 2014.

[12] K. Deb, A. Pratap, S. Agarwal and T. A. M. T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II." *IEEE Trans. Evolut. Comput.*, vol. 6, no. 2, pp. 182-197, Apr. 2002.

[13] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1628–1656, 2017.

[14] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," *IEEE Int. Symp. Inform. Theory (ISIT)*, Barcelona, Spain, pp. 1451–1455, 2016.

[15] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

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

[17] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no.8, pp. 7432–7445, Aug. 2017.

[18] M. Deng, H. Tian, and X. Lyu, "Adaptive sequential offloading game for multi-cell mobile edge computing," in *23rd Int. Conf. Telecommun. (ICT)*, pp. 1–5, 2016.

[19] J. Zhang, X. Hu, Z. Ning, E. C. H. Ngai, L. Zhou, J. Wei, J. Cheng and B. Hu. "Energy-latency trade-off for energy-aware offloading in mobile edge computing networks." *IEEE Internet Things J.*, pp (99). 1-1, 2017.

[20] Y. Mao, J. Zhang and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Area. Comm.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[21] S. Bi and Y. J. Zhang. "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading." *IEEE Trans. Wirel. Commun.*, vol. 17, no. 6, pp. 4177-4190, Jun. 2018.

[22] L. Liu, Z. Chang, X. Guo, S. Mao and T. Ristaniemi. "Multi-objective optimization for computation offloading in fog computing." *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283-294, Feb. 2018.

[23] H. Huang and S. Guo, "Adaptive service provisioning for mobile edge cloud." *ZTE Commun.*, vol. 15, no. 2, pp.1-9, 2017.

[24] H. Huang and S. Guo. "Service provisioning update scheme for mobile application users in a cloudlet network." *IEEE International Conference on Communications (ICC)*, 2017.

[25] J. Wu, S. Guo, J. Li and D. Zeng. "Big data meet green challenges: big data toward green applications." *IEEE Syst. J.*, vol. 10, no. 3, pp. 888-900, Sep. 2016.

[26] J. Wu, S. Guo, J. Li and D. Zeng. "Big data meet green challenges: Greening big data." *IEEE Syst. J.*, vol. 10, no. 3, pp. 873-887, Sep. 2016.

[27] J. Wu, S. Guo, H. Huang, W. Liu and Y. Xiang. "Information and Communications Technologies for Sustainable Development Goals: State-of-the-Art, Needs and Perspectives." *IEEE Commun. Surv. Tut.*, vol. 20, no. 3, pp. 2389-2406, 2018.

[28] K. Wang, Y. Wang, Y. Sun, S. Guo and J. Wu. "Green industrial internet of things architecture: An energy-efficient perspective." *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 48-54, Dec. 2016.

[29] A. H. Jafari, D. Lpez-Prez, H. Song, H. Claussen, L. Ho, and J. Zhang, "Small cell backhaul: challenges and prospective solutions," *Eurasip J. Wirel. Comm.*, vol. 2015, no. 1, pp. 206, Aug. 2015.

[30] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct. 2017.

[31] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2827-2840, Oct. 2016.

[32] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[33] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in Proc. *IEEE 7th Inter. Symp. Service Orient. Syst. Eng.*, Redwood City, CA, USA, pp. 494–502, Mar. 2013.

[34] H. A. Mostafa, R. El-Shatshat and M. M. A. Salama. "Multi-Objective Optimization for the Operation of an Electric Distribution System With a Large Number of Single Phase Solar Generators," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 1038-1047, Jun. 2013.

[35] Q. Zhang and Hui Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Trans. Evolut. Comput.*, vol.11, no. 6, pp. 712-731, Dec. 2007.

[36] E. C. Jara, "Multi-Objective Optimization by Using Evolutionary Algorithms: The p-Optimality Criteria," *IEEE Trans. Evolut. Comput.*, vol. 18, no. 2, pp. 167- 179, Apr. 2014.

[37] C. Peng, H. Liu and F. Gu, "An evolutionary algorithm with directed weights for constrained multi-objective optimization," *Appl. Soft Comput.*, vol. 60, pp. 613–622, Jul. 2017.

[38] X. Niu, H. Wang, S. Hu, C. Yang, and Y. Wang, "Multi-objective online optimization of a marine diesel engine using nsga-ii coupled with enhancing trained support vector machine." Appl. Therm. Eng., vol. 137, pp. 218-227, Jun. 2018.

[39] S. S. Miriyala, V. Subramanian and K. Mitra, "Transform-ann for online optimization of complex industrial processes: casting process as case study." Eur. J. Oper. Res., vol 264, no. 1, pp. 294-309, Jan. 2018.

[40] N. Rehani and R. Garg. "Meta-heuristic based reliable and green workflow scheduling in cloud computing." Int. J. Syst. Assur. Eng. Manag., vol. 9, no. 4, pp. 811-820, Aug. 2018.