Evolutionary Large-scale Sparse Multi-objective Optimization for Collaborative Edge-cloud Computation Offloading

Guang Peng[1], Huaming Wu[2], Han Wu[1] and Katinka Wolter[1]

[1]*Department of Mathematics and Computer Science, Free University of Berlin, Takustr. 9, Berlin, Germany*

[2]*Center for Applied Mathematics, Tianjin University, Tianjin 300072, China*

Keywords: Local-edge-cloud, Computation Offloading, Large-scale Multi-objective Optimization, Restricted Boltzmann Machine, Contribution Score.

Abstract: This paper proposes evolutionary large-scale sparse multi-objective optimization (ELSMO) algorithms for collaboratively solving edge-cloud computation offloading problems. To begin with, a collaborative edge-cloud computation offloading multi-objective optimization model is established in a mobile environment, where the offloading decision is represented as a binary encoding. Considering the large-scale and sparsity property of the computation offloading model, the restricted Boltzmann machine (RBM) is applied to reduce the dimensionality and learn the Pareto-optimal subspace. In addition, the contribution score of each decision variable is assumed to generate new offsprings. Combining the RBM and the contribution score, two evolutionary algorithms using non-dominated sorting and crowding distance methods are designed, respectively. The proposed algorithms are compared with other state-of-the-art algorithms and offloading strategies on a number of test problems with different scales. The experiment results demonstrate the superiority of the proposed algorithms.

1 INTRODUCTION

With the development of mobile networks and the Internet of Things (IoT), more and more complicated and time-sensitive applications are being developed for mobile devices (MDs), e.g., face recognition, augmented reality and interactive gaming. Due to the insufficient computing ability as well as the limited battery power of MDs, the quality of service (QoS) of these resource-intensive and delay-sensitive applications cannot be satisfied if they are handled locally. Fortunately, mobile cloud computing (MCC) (Sahu and Pandey, 2018) and mobile edge computing (MEC) (Mach and Becvar, 2017) have emerged for solving these problems, where the complex computing tasks can be chosen to be offloaded to a central cloud or an edge cloud.

In MCC, taking advantage of more powerful computing capability in a cloud data center, the resource-intensive tasks can be offloaded to a central cloud to alleviate the limitation of computing capability in MDs. Considering the central cloud may be a little far away from MDs, it may cost more time for transferring the tasks. MEC enables MDs to offload tasks to servers located at the edge of the cellular network, which reduces the latency of the communication between MDs and edge servers. Wu (Wu et al., 2019) proposed a min-cost offloading partitioning (MCOP) algorithm inspired by graph cutting to minimize the weighted sum of time and energy in MCC/MEC. Based on his work, Sheikh (Sheikh and Das, 2018) modeled the effect of parallel execution on multi-site computation offloading in MCC and used an existing genetic algorithm to solve the problem. Their work focuses on the single-objective optimization of offloading. Some other work used deep learning methods to solve offloading problems. Huang (Huang et al., 2018) formulated the joint offloading decision and bandwidth allocation as a mixed integer programming problem in MEC, and proposed a distributed deep learning-based offloading (DDLO) algorithm for generating offloading decisions. Yu (Yu et al., 2020) devised a new deep imitation learning (DIL)-driven edge-cloud computation offloading framework for minimizing the time cost in MEC networks. For deep learning methods, one important concern is that the neural network may need a lot of time to be trained, and the changing parameters in a mobile environment can change the structure of the network. On the other hand, multiple meta-heuristic optimization algorithms have also gathered attention. Xu (Xu et al., 2019) applied the evolutionary algorithm NSGA-III (non-dominated sorting genetic algorithm III) to deal with computation offloading over

Peng, G., Wu, H., Wu, H. and Wolter, K.
Evolutionary Large-scale Sparse Multi-objective Optimization for Collaborative Edge-cloud Computation Offloading.
DOI: 10.5220/0010145501000111
In *Proceedings of the 12th International Joint Conference on Computational Intelligence (IJCCI 2020)*, pages 100-111
ISBN: 978-989-758-475-6

big data for IoT-enabled cloud-edge computing. Guo (Guo et al., 2018) designed a hierarchical genetic algorithm (GA) and particle swarm optimization (PSO) algorithm to solve the energy-efficient computation offloading management problem.

However, the above methods especially traditional evolutionary algorithms will encounter difficulties to deal with large-scale computation offloading problems when there exists a large number of MDs having a huge number of computation tasks. It is difficult for the traditional evolutionary algorithms to search in the whole high-dimensional space with limited population size and computational resources. In this paper, the proposed computation offloading model is treated as a multi-objective optimization problem (MOP) and we focus on multi-objective optimization evolutionary algorithms (MOEAs). For solving large-scale MOPs, LMEA (Zhang et al., 2018) divided the decision variables into convergence-related and diversity-related variables and optimizes them by different strategies. LCSA (Zille and Mostaghim, 2019) used a linear combination of population members to tackle large-scale MOPs. LSMOF (He et al., 2019) applies two reference points on a solution to search for better solutions in the large-scale search space. Although these MOEAs are tailored for large-scale MOPs, they cannot be applied to MOPs with binary decision variables and they are shown to be of low efficiency with a large number of function evaluations (Tian et al., 2020b). Considering some large-scale MOPs whose Pareto optimal solutions are sparse, the literature (Tian et al., 2020a) used two unsupervised neural networks, a restricted Boltzmann machine, and a denoising autoencoder to learn a sparse distribution and compact representation of the decision variables, which is regarded as an approximation of the Pareto-optimal subspace. In literature (Tian et al., 2020b), a hybrid representation of solution is adopted to integrate real and binary encodings, which can solve both large-scale sparse benchmarks and four real applications well.

Following the above ideas, in this paper we set up a collaborative edge-cloud computation offloading large-scale sparse multi-objective optimization model with binary encoding. Inspired from literature (Tian et al., 2020a) and (Tian et al., 2020b), based on the dimensionality reduction and decision variable analysis methods, two evolutionary large-scale sparse multi-objective optimization algorithms are proposed and compared to solve the large-scale offloading problems. Compared with other MOEAs and offloading schemes, the proposed algorithms are competitive on different large-scale test problems.

The rest of this paper is organized as follows. Section 2 briefly introduces the brief background. The local-edge-cloud offloading multi-objective optimization model is presented in Section 3. The details of the proposed algorithms are described in Section . The experimental studies are discussed in Section 4. Finally, Section 6 summarizes and presents the conclusion and future work.

2 BACKGROUND

In this section some background concepts are provided.

2.1 Multi-objective Optimization

A multi-objective optimization problem (MOP) can be defined as follows:

$$\begin{aligned} \min\ & F(x) = \left(f_1(x), f_2(x), \cdots, f_m(x)\right)^T \\ & \text{subject to } x \in \Omega \subseteq R^n, \end{aligned} \tag{1}$$

where Ω is the decision space and x is a solution; $F : \Omega \to \Theta \subseteq R^m$ denotes the m-dimensional objective vector and Θ is the objective space.

A solution x^0 is said to Pareto dominate another solution x^1, denoted by $x^0 \prec x^1$, if

$$\begin{cases} f_i\left(x^0\right) \leq f_i\left(x^1\right), & \forall i \in \{1,2,\cdots,m\} \\ f_j\left(x^0\right) < f_j\left(x^1\right), & \exists j \in \{1,2,\cdots,m\}. \end{cases} \tag{2}$$

A solution x^0 is called a Pareto optimal solution, if $\neg\exists x^1 : x^1 \prec x^0$.

The set of Pareto optimal solutions is defined as $PS = \left\{x^0 \left| \neg\exists x^1 \prec x^0 \right.\right\}$.

The Pareto optimal solution set in the objective space is called Pareto front (PF).

2.2 Restricted Boltzmann Machine

Restricted Boltzmann Machine (RBM) is a stochastic neural network, which consists of an input layer and a hidden layer, as shown in Fig. 1. The nodes in the two layers are binary variables obeying binomial distribution. RBM can be used to reduce the dimensionality through unsupervised learning. Given an input vector x, the value of each node h_j in the hidden layer is set to 1 with a probability:

$$p\left(h_j = 1 \left| x \right.\right) = \sigma\left(a_j + \sum_{i=1}^{D} x_i w_{ij}\right) \tag{3}$$

where a_j is the basis, w_{ij} is the weight, D is the dimensionality of input layer, $\sigma(x) = 1/\left(1 + \exp(-x)\right)$ is the sigmoid function. Through comparing the probability $p\left(h_j = 1 \left| x \right.\right)$ with a uniformly distributed random value in $[0,1]$, the binary value of each node h_j

can be obtained. In the same way, the reconstructed value of each node x'_i in the input layer is set to 1 with a probability:

$$p\left(x'_i = 1 | h\right) = \sigma\left(a'_j + \sum_{j=1}^{K} h_j w_{ij}\right) \tag{4}$$

where K is the dimensionality of hidden layer. Hinton (Hinton, 2002) proposed the contrastive divergence algorithm to train RBM, which aims to minimize the reconstruction error between the reconstructed vector x' and the original input x by finding the suitable a, a', and w.

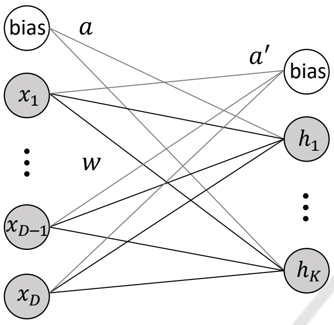


Figure 1: RBM structure.

3 SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we consider a collaborative MEC and MCC network with one edge cloud, one central cloud and multiple mobile devices (MDs). Thus, the mobile device can execute its computational tasks locally, or offload its tasks to the edge cloud server through a wireless link and/or to the central cloud server through wireless and backhaul links.

3.1 System Model

Fig. 2 presents the system model, which consists of one edge cloud server, one central cloud server and multiple MDs, denoted by a set $\mathcal{N} = \{1,2,\ldots,N\}$. The edge cloud server can be deployed into the base station, which is closer to the MDs. The MDs can communicate with the edge cloud with a wireless link, whereas the edge cloud and the central cloud can be interconnected through a wired link. Each MD has multiple tasks, denoted by a set $\mathcal{M} = \{1,2,\cdots,M\}$. The size of m-th task of the n-th MD is denoted by $w_{(n,m)}$. In each MD, these different tasks can choose to be processed locally or offloaded to the edge cloud server and the central cloud server. The offloading decision is represented by two binary variables $x^1_{(n,m)}$ and $x^2_{(n,m)}$.

On one hand, $x^1_{(n,m)} \in \{0,1\}$ denotes the offloading decision for m-th task, which means:

$$x^1_{(n,m)} = \begin{cases} 0, & \text{if task is executed locally} \\ 1, & \text{if task is offloaded} \end{cases} \tag{5}$$

where $x^1_{(n,m)} = 0$ denotes n-th MD decides to execute the m-th task locally, $x^1_{(n,m)} = 1$ indicates n-th MD decides to offload m-th task to the edge cloud server or central cloud server.

Once the m-th task is decided to be offloaded to the cloud server, $x^2_{(n,m)} \in \{0,1\}$ represents the specific offloading destination for the m-th task, which means:

$$x^2_{(n,m)} = \begin{cases} 0, & \text{if offloaded to edge cloud } \& \; x^1_{(n,m)} = 1 \\ 1, & \text{if offloaded to central cloud } \& \; x^1_{(n,m)} = 1 \end{cases} \tag{6}$$

where $x^2_{(n,m)} = 0$ denotes n-th MD decides to offload m-th task to the edge cloud server, $x^2_{(n,m)} = 1$ denotes m-th task is offloaded to the central cloud server.

The detailed operations of local computing, edge cloud computing and central cloud computing are illustrated in Sections 3.2, 3.3 and 3.4, respectively. The important notations used in this paper are listed in Table 1.

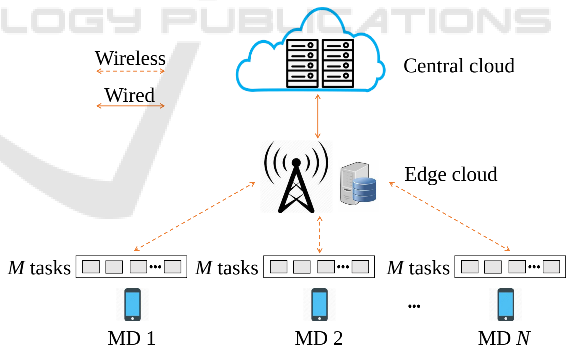


Figure 2: System model of computation offloading with heterogeneous cloud.

3.2 Local Computing Model

We first establish the local computing model when the task is decided to be executed locally. The task execution time of mobile device can be calculated as:

$$Tl_{(n,m)} = \frac{w_{(n,m)}}{f_l} \tag{7}$$

where f_l denotes the task processing rate of local device.

Table 1: Important notations used in this paper.

Notation	Description
$w_{(n,m)}$	The m-th task workload of the n-th MD
$x^1_{(n,m)}$	$x^1_{(n,m)} = 0$ if m-th task is executed locally, $x^1_{(n,m)} = 1$ if m-th task is offloaded to the cloud
$x^2_{(n,m)}$	$x^2_{(n,m)} = 0$ if m-th task is offloaded to edge cloud, $x^2_{(n,m)} = 1$ if m-th task is offloaded to central cloud
$El_{(n,m)}$	The energy consumption of the m-th task of the n-th MD
θ_l	The local device energy consumption per unit of workload
$Tl_{(n,m)}$	The execution time of the m-th task of the n-th MD
f_l	The task processing rate of the MD
$T^e_{t\,(n,m)}$	The transmission time of offloading m-th task to edge cloud via wireless link
$b_{(n,e)}$	The bandwidth between n-th MD and edge cloud
$E^e_{t\,(n,m)}$	The energy consumption for transmission to the edge cloud
σ	The energy consumption per unit of workload for transmission to the edge cloud
$Te_{(n,m)}$	The time delay of offloading the m-th task of the n-th MD to the edge cloud
f_e	The task processing rate of the edge cloud
$Ee_{(n,m)}$	The energy consumption of offloading the m-th task of the n-th MD to the edge cloud
θ_e	The energy consumption per unit of workload of edge cloud
$T^c_{t\,(n,m)}$	The transmission time of offloading m-th task to central cloud via wireless link and wired link
$b_{(e,c)}$	The bandwidth between edge cloud and central cloud
$E^c_{t\,(n,m)}$	The energy consumption for transmission to the central cloud
β	The energy consumption per unit of workload for transmission to the central cloud
$Tc_{(n,m)}$	The time delay of offloading the m-th task of the n-th MD to the central cloud
f_c	The task processing rate of the central cloud
$Ec_{(n,m)}$	The energy consumption of offloading the m-th task of the n-th MD to the central cloud
θ_c	The energy consumption per unit of workload of central cloud

The energy consumption for executing m-task at the n-th device can be calculated as:

$$El_{(n,m)} = \theta_l w_{(n,m)} \tag{8}$$

Where θ_l denotes the energy consumption per unit of workload of local device.

Therefore, the total computation time and energy consumption of the n-th MD can be expressed as:

$$Tl_{(n)} = \sum_{m=1}^{M} \left(1 - x^1_{(n,m)}\right) Tl_{(n,m)} \tag{9}$$

$$El_{(n)} = \sum_{m=1}^{M} \left(1 - x^1_{(n,m)}\right) El_{(n,m)} \tag{10}$$

3.3 Edge Cloud Computing Model

For the edge cloud computing model, the MDs can communicate with the edge cloud via the cellular link. When the task is decided to be offloaded to the edge cloud, the MD needs to transmit the workload of the task to the edge cloud and then to be processed. In general, the time and energy consumption are often neglected when the cloud servers return the computing results back to MDs, because the data size of feedback result is small (Bi and Zhang, 2018).

The transmission time for offloading the workload to the edge cloud server can be calculated as:

$$T^e_{t\,(n,m)} = \frac{w_{(n,m)}}{b_{(n,e)}} \tag{11}$$

where $b_{(n,e)}$ denotes the bandwidth between n-th MD and edge cloud.

The energy consumption for the transmission can be given by:

$$E^e_{t\,(n,m)} = \sigma w(n,m) \tag{12}$$

where σ denotes the energy consumption per unit of workload for transmission to the edge cloud server.

After the task is transmitted to the edge cloud, it will be executed at the edge cloud server. The computation delay of the whole process can be expressed as:

$$Te_{(n,m)} = T^e_{t\,(n,m)} + \frac{w_{(n,m)}}{f_e} \tag{13}$$

where f_e denotes the task processing rate of the edge cloud server.

The energy consumption of whole process can be expressed as:

$$Ee_{(n,m)} = E^e_{t\,(n,m)} + \theta_e w_{(n,m)} \tag{14}$$

where θ_e denotes the energy consumption per unit of workload of edge cloud.

Therefore, the total computation time and energy consumption of the n-th MD can be expressed as:

$$Te_{(n)} = \sum_{m=1}^{M} x^1_{(n,m)} \left(1 - x^2_{(n,m)}\right) Te_{(n,m)} \tag{15}$$

$$Ee_{(n)} = \sum_{m=1}^{M} x^1_{(n,m)} \left(1 - x^2_{(n,m)}\right) Ee_{(n,m)} \tag{16}$$

3.4 Central Cloud Computing Model

For the central cloud computing model, MDs can communicate with the central cloud via wireless and wired links. The central cloud servers can provide more powerful computing capacity for the MDs, but it might cause more delays for the transmission between MDs and the central cloud. The transmission time and energy consumption for offloading the workload to the central cloud server can be calculated as:

$$T^c_{t\,(n,m)} = \frac{w_{(n,m)}}{b_{(n,e)}} + \frac{w_{(n,m)}}{b_{(e,c)}} \tag{17}$$

$$E^c_{t\,(n,m)} = \sigma w(n,m) + \beta w(n,m) \tag{18}$$

where $b_{(e,c)}$ denotes the bandwidth between edge cloud and central cloud. β denotes the energy consumption per unit of workload for transmission to the central cloud server.

After the task is transmitted to the central cloud, it will be executed at the central cloud server. The computation delay and energy consumption of the whole process can be expressed as:

$$Tc_{(n,m)} = T^c_{t\,(n,m)} + \frac{w_{(n,m)}}{f_c} \tag{19}$$

$$Ec_{(n,m)} = E^c_{t\,(n,m)} + \theta_c w_{(n,m)} \tag{20}$$

where f_c denotes the task processing rate of central cloud server, θ_c denotes the energy consumption per unit of workload of central cloud.

Therefore, the total computation time and energy consumption of the n-th MD can be expressed as:

$$Tc_{(n)} = \sum_{m=1}^{M} x^1_{(n,m)} x^2_{(n,m)} Tc_{(n,m)} \tag{21}$$

$$Ec_{(n)} = \sum_{m=1}^{M} x^1_{(n,m)} x^2_{(n,m)} Ec_{(n,m)} \tag{22}$$

3.5 Problem Formulation

The total computation time of executing all tasks can be given by:

$$T = \sum_{n=1}^{N} \max\left\{Tl_{(n)}, Te_{(n)}, Tc_{(n)}\right\} \tag{23}$$

The total energy consumption of executing all tasks can be given by:

$$E = \sum_{n=1}^{N} \left(El_{(n)} + Ee_{(n)} + Ec_{(n)}\right) \tag{24}$$

To summarize, we establish a local-edge-cloud computation offloading two-objective optimization model. Considering a large number of mobile devices having different applications in the mobile environment, the offloading model is the large-scale MOP.

4 THE PROPOSED ALGORITHM

This section presents the details of two proposed algorithms for solving the large-scale computation offloading problems.

4.1 The General Framework

The general frameworks of two proposed algorithms are presented in Algorithm 1 and 2, respectively. In both two algorithms, the non-dominated front number and crowding distance are calculated as the same way in NSGA-II (Deb et al., 2002), which are also the selection criteria in the environmental selection. In ELSMO-1, the non-dominated solutions in the current population are used to train a RBM, then the offspring solutions are generated from the mating pool. The two parameters ρ and K are updated iteratively. In ELSMO-2, the population initialization is different from ELSMO-1, which analyzes the contribution score of each decision variable. And the new offspring generation operation is conducted based on the score. More details about the main operations in ELSMO-1 and ELSMO-2 are presented in the following sections.

4.2 The Proposed ELSMO-1

4.2.1 Offspring Generation

Before generating offspring solutions, all the non-dominated solutions in the population are used to train a RBM via the contrastive divergence algorithm. As shown in Fig. 3, the original binary vectors can be reduced the dimensionality by RBM, and the reduced binary vectors can be also recovered. Since the non-dominated solutions are used to train the RBM, it can be seen that each solution can be mapped between the Pareto optimal space and the original search space.

After obtaining the trained RBM, the binary tournament selection mechanism is used to select $\widetilde{N}$ parents as the mating pool based on the non-dominated

Algorithm 1: Framework of ELSMO-1.

```
    Input: The population size $\widetilde{N}$
    Output: The final population $P$
 1  $P \leftarrow Initialization\left(\widetilde{N}\right)$;
 2  $[F_1, F_2, \cdots] \leftarrow NondominatedSorting(P)$;
 3  $CrowdDis \leftarrow CrowdingDistance(F_1, F_2, \cdots)$;
 4  $\rho \leftarrow 0.5$; // Ratio of offspring solutions
      generated in the different search space
 5  $K \leftarrow N$; // Size of the hidden layer
 6  while $termination\ criterion\ not\ fulfilled$ do
 7      Train a RBM with $K$ hidden neurons based
          on non-dominated solutions in $P$;
 8      $P' \leftarrow$ Select $\widetilde{N}$ parents via binary
          tournament selection in $P$ ;
 9      $O \leftarrow$
          $OffspringGeneration(P, P', \rho, K, RBM)$;
10      $P \leftarrow P \cup O$;
11      Delete duplicated solutions from $P$;
12      $P \leftarrow EnvironmentalSelection(P)$;
13      $[\rho, K] \leftarrow UpdateParameter(P, \rho)$ ;
14  end
```

Algorithm 2: Framework of ELSMO-2.

```
    Input: The population size $\widetilde{N}$
    Output: The final population $P$
 1  $[P, Score] \leftarrow Initialization\left(\widetilde{N}\right)$;
 2  $[F_1, F_2, \cdots] \leftarrow NondominatedSorting(P)$;
 3  $CrowdDis \leftarrow CrowdingDistance(F_1, F_2, \cdots)$;
 4  while $termination\ criterion\ not\ fulfilled$ do
 5      $P' \leftarrow$ Select $2\widetilde{N}$ parents via binary
          tournament selection in $P$ ;
 6      $O \leftarrow OffspringGeneration(P', Score)$;
 7      $P \leftarrow P \cup O$;
 8      Delete duplicated solutions from $P$;
 9      $P \leftarrow EnvironmentalSelection(P)$;
10  end
```

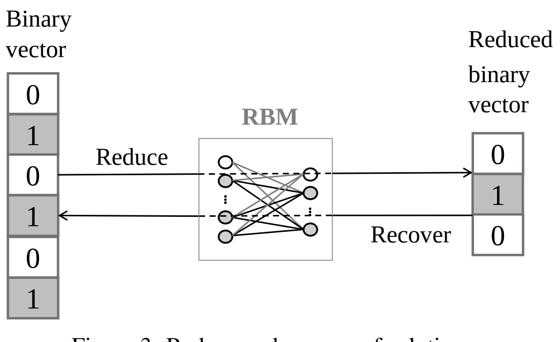


Figure 3: Reduce and recover of solutions.

front number and crowding distance. Then two parents are randomly selected from the mating pool and single-point crossover and bitwise mutation operation are used to generate two offspring solutions. The ratio ρ determines the probability that the offspring solutions will be generated in the Pareto-optimal subspace by the trained RBM or generated in the original search space. Specifically, if the probability ρ is larger than a random value in $[0, 1]$, the binary vectors of parents are reduced by Eq. (3) and the offspring solutions are generated in the Pareto-optimal space, then the reduced offspring solutions are recovered by Eq. (4). If the probability ρ is smaller than the random value, the offspring solutions are generated in the original search space without RBM.

4.2.2 Update Parameter

In ELSMO-1, there are two parameters needed to be considered, the ratio of offspring solutions generated in the Pareto-optimal subspace or original search space ρ and the size of the hidden layer K. The parameter ρ is updated iteratively, which is defined as follows:

$$\rho_{t+1} = 0.5 \times \left(\rho_t + \frac{s_{1,t}}{s_{1,t} + s_{2,t}}\right) \tag{25}$$

where ρ_t is the value of ρ at the t-th generation and $\rho_0 = 0.5$. $s_{1,t}$ and $s_{2,t}$ denotes the number of successful offspring solutions generated in the Pareto-optimal subspace and in the original search space, respectively. A successful solution means that it survives to the next population.

The sparsity of non-dominated solutions in the current population reflects the setting value of K. Let dec be a binary vector (the same dimensionality to offloading problem dimension D) denoting whether each variable should be nonzero, the probability of setting dec_i to 1 is defined according to the non-dominated solution set NP:

$$p(dec_i = 1 | NP) = \frac{1}{|NP|} \sum_{x \in NP} |sign(x_i)| \tag{26}$$

where if $x_i = 0$ then $|sign(x_i)|$ equals 0 or $|sign(x_i)|$ is equal to 1 otherwise. Through comparing the probability $p(dec_i = 1 | NP)$ with a uniformly distributed random value in $[0, 1]$, the value of dec_i is obtained. Then the parameter K is defined as follows:

$$K = \sum dec_i \tag{27}$$

4.3 The Proposed ELSMO-2

4.3.1 The Population Initialization

In ELSMO-2, the population initialization process includes two steps, i.e., calculating the scores of decision variables and generating the initial population. First, the population Q is constituted by a $D*D$ identity matrix, where D denotes the number of decision variables. Then the non-dominated front numbers of the solutions in population Q are calculated based on the non-dominated sorting method. The non-dominated front number of the i-th solution in Q can be regarded as the score of the i-th decision variable. The higher score means the better quality of the decision variable, so the value of the decision variable is set to 1 with higher probability.

Afterwards, let a binary vector *mask* represent the offloading decision. According to the scores of decision variables, the binary tournament selection mechanism is used to select the element in *mask* with a better score and set the element to 1. In each solution, the number of $rand\left(\right) \times D$ elements are selected to be set to 1 and other elements in *mask* are set to 0, where $rand\left(\right)$ denotes a uniformly distributed random value in $[0,1]$. In this way, the initialized population with better convergence and diversity is obtained by selecting decision variables with good quality.

4.3.2 The Offspring Generation

The binary tournament selection mechanism is used to select $2\widetilde{N}$ parents as the mating pool based on the non-dominated front number and crowding distance. Then two parents p and q are randomly selected from the mating pool to generate an offspring o each time. The binary vector *mask* of o is first set to the same to p, then two decision variables from the nonzero elements in $p.mask \cap \overline{q.mask}$ are randomly selected with probability 0.5, the decision variable in the *mask* of o with a larger score is set to 0. Otherwise, selecting two decision variables from nonzero elements in $\overline{p.mask} \cap q.mask$, the decision variable in the *mask* of o with smaller score is set to 1.

Afterwards, one mutation operation is conducted on the *mask* of o to retain diversity. Compare one random probability distributed in $[0,1]$ with 0.5, if the probability is less than 0.5, randomly select two decision variables from the nonzero elements in $o.mask$, and set the element with large score in $o.mask$ to 0. Otherwise, randomly select two decision variables from the nonzero elements in $\overline{o.mask}$, and set the element with a small score in $o.mask$ to 1. The main idea of the mutation operation is making decision variables with better quality approach to 1, while decision variables with worse quality approach 0.

4.4 Computational Complexity

For the proposed algorithms, the major costs are the iteration process in Algorithm 1 and 2. In ELSMO-1, Step 7 needs $O\left(\widetilde{N}EDK\right)$ operations to train the RBM, where $\widetilde{N}$ is the population size, E is the number of epochs for training, D is the number of decision variables, K is the hidden layer size. Step 8 needs $O\left(\widetilde{N}\right)$ operations for the binary tournament selection. Step 9 performs $O\left(\widetilde{N}DK\right)$ to generate offspring solutions. Step 12 performs $O\left(\widetilde{M}\widetilde{N}^2\right)$ operations for the environmental selection, where $\widetilde{M}$ is the number of objectives. Step 13 needs $O\left(\widetilde{N}D\right)$ operations to update the parameters. To summarize, the overall computational complexity at one generation of ELSMO-1 is $\max\left\{O\left(\widetilde{N}EDK\right), O\left(\widetilde{M}\widetilde{N}^2\right)\right\}$. In ELSMO-2, Step 5 performs $O\left(2\widetilde{N}\right)$ operations for selecting mating pool. Step 6 needs $O\left(\widetilde{N}D\right)$ operations to generate offspring solutions. Step 9 performs $O\left(\widetilde{M}\widetilde{N}^2\right)$ operations for the environmental selection. The overall computational complexity at one generation of ELSMO-2 is $\max\left\{O\left(\widetilde{M}\widetilde{N}^2\right), O\left(\widetilde{N}D\right)\right\}$.

NSGA-II (Deb et al., 2002), SPEA2 (Zitzler et al., 2001), SMS-EMOA (Hochstrate et al., 2007), and EAG-MOEA/D (Cai et al., 2015) are four compared algorithms selected in this paper. The computational complexity of NSGA-II, SMS-EMOA and EAG-MOEA/D is the same, i.e. $O\left(\widetilde{M}\widetilde{N}^2\right)$. The worst computational complexity of SPEA2 is $O\left(\left(\widetilde{N}+\overline{N}\right)^3\right)$, where $\overline{N}$ is the archive size.

5 EXPERIMENTAL STUDIES

In this section, we evaluate the performance of the proposed algorithms and demonstrate the compared results of different MOEAs and offloading strategies.

5.1 Experimental Settings

In the experiment, we set up the local-edge-cloud offloading environment. The number of mobile devices is selected between 100 and 1000. The number of independent tasks of each MD is $M = 5$. So the dimension of the offloading problem D is between 1000 and 10000. We set the energy consumption per unit of workloads in local device, edge and central cloud server $\theta_l = 3\text{J/MB}$, $\theta_e = 1.5\text{J/MB}$, and $\theta_c = 1\text{J/MB}$, respectively. The processing rates of local device, edge and central cloud server are $f_l = 2\text{MB/s}$, $f_e = 8\text{MB/s}$, and $f_c = 12\text{MB/s}$, respectively. The energy consumption per unit of workloads for transmission from local device to edge cloud server is a random value within $[0.4, 0.6]$ J/MB, and the same measure from edge to central cloud server is random value within $[0.3, 0.5]$ J/MB. In addition, the bandwidth between the local device and edge server is chosen from $[80, 100]$ Mbps, whereas the bandwidth between edge cloud and central cloud is fixed $b_{(e,c)} = 150\text{Mbps}$. We assume that the workloads of tasks are randomly distributed between 10MB and 30MB. The related parameters and corresponding values are summarized in Table 2.

To verify the performance of the proposed algorithms, we compare the proposed algorithms with other MOEAs and offloading schemes to solve ten different large-scale offloading problems, which means that the number of devices N = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000] and the dimension D = [1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000]. The compared algorithms are NSGA-II (Deb et al., 2002), SPEA2 (Zitzler et al., 2001), SMS-EMOA (Hochstrate et al., 2007), and EAG-MOEA/D (Cai et al., 2015). NSGA-II, SPEA2, SMS-EMOA are three classical MOEAs which are effective for MOPs, while EAG-MOEA/D is tailored for combinatorial MOPs. For a fair comparison, the population size of all algorithms is set to 50. The number of function evaluations is set with values in the interval from 6.0×10^4 to 15×10^4 for ten test problems, and the interval of the number of function evaluations is 10^4. In NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D and ELSMO-1, the single-point crossover and bitwise mutation are applied to generate new offspring, where the probabilities of crossover and mutation are set to 1.0 and $1/D$, respectively. The hypervolume (HV) (Zitzler et al., 2003) is adopted as the metric to evaluate the performance of the compared algorithms. The HV can reflect the performance regarding both convergence and diversity. The bigger the HV value, the better the performance of the algorithm. For each test instance, each algorithm is executed 30 times independently, and the average and standard deviation of the 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 compared algorithm is significantly better, significantly worse and similar to that obtained by the proposed algorithm.

The other four compared offloading schemes are Local Offloading Scheme (LOS), Edge Offloading Scheme (EOS), Cloud Offloading Scheme (COS), Random Offloading Scheme (ROS). LOS, EOS, and COS represent that all applications are executed on the local device, offloaded to edge and central cloud. ROS denotes that the offloading decisions are generated randomly. To demonstrate the effectiveness of different offloading schemes, the offloading gain of a weighted sum of time and energy based on LOS is defined as:

$$OffloadingGain = \Big[w \times \frac{T_{LOS} - T_{offloading}}{T_{LOS}} + (1-w) \times \frac{E_{LOS} - E_{offloading}}{E_{LOS}} \Big] \times 100\% \tag{28}$$

where T_{LOS} and E_{LOS} denote the time and energy consumption of LOS, respectively. $T_{offloading}$ and $E_{offloading}$ denotes the time and energy consumption of other offloading schemes. w is the weight parameter, which can be set by the decision-maker.

5.2 Comparison with Other MOEAs

Table 3 presents the HV metric values obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on ten test problems. The proposed ELSMO-2 has achieved the best performance on 9 of 10 test instances, while only EAG-MOEA/D gets 1 of 10 best results for the rest of compared algorithms. When the dimension is not so large (i.e., 1000), the EAG-MOEA/D and ELSMO-2 can obtain similar metric values. It can be observed that ELSMO-2 has a clear advantage over other compared algorithms with the increment of dimension.

Fig. 4, Fig. 5 and Fig. 6 show the final non-dominated solution set with the medium HV value obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on offloading problems with 1000, 5000, and 10000 binary variables. NSGA-II, SPEA2, SMS-EMOA can only get a small part of the solutions in the Pareto front, which may be worse when dealing with large-dimension problems. EAG-MOEA/D is designed for combinatorial problems, which can obtain good performance compared with other classical algorithms NSGA-II, SPEA2 and SMS-EMOA, whereas its diversity still encounters difficulties for solving large-dimensional offloading problems. ELSMO-1 seems to have the best performance of diversity compared with the other five algorithms, but the RBM may need more iterations to be trained for improving the convergence. It

Table 2: Parameter values.

Parameter	Value
The number of mobile devices	$N = [100, 1000]$
The number of independent tasks of each MD	$M = 5$
The local energy consumption per unit	$\theta_l = 3\text{J/MB}$
The edge cloud energy consumption per unit	$\theta_e = 1.5\text{J/MB}$
The central cloud energy consumption per unit	$\theta_c = 1\text{J/MB}$
The processing rate of the local device	$f_l = 2\text{MB/s}$
The processing rate of the edge cloud server	$f_e = 8\text{MB/s}$
The processing rate of the central cloud server	$f_c = 12\text{MB/s}$
The energy consumption per unit for transmission from MD to edge cloud	$\sigma = [0.4, 0.6]\,\text{J/MB}$
The energy consumption per unit for transmission from edge to central cloud	$\beta = [0.3, 0.5]\,\text{J/MB}$
The bandwidth between n-th MD and edge cloud	$b_{(n,e)} \in [80, 100]\,\text{Mbps}$
The bandwidth between edge and central cloud	$b_{(e,c)} = 150\text{Mbps}$
The workloads of all tasks	$w_{(n,m)} \in [10, 30]\,\text{MB}$

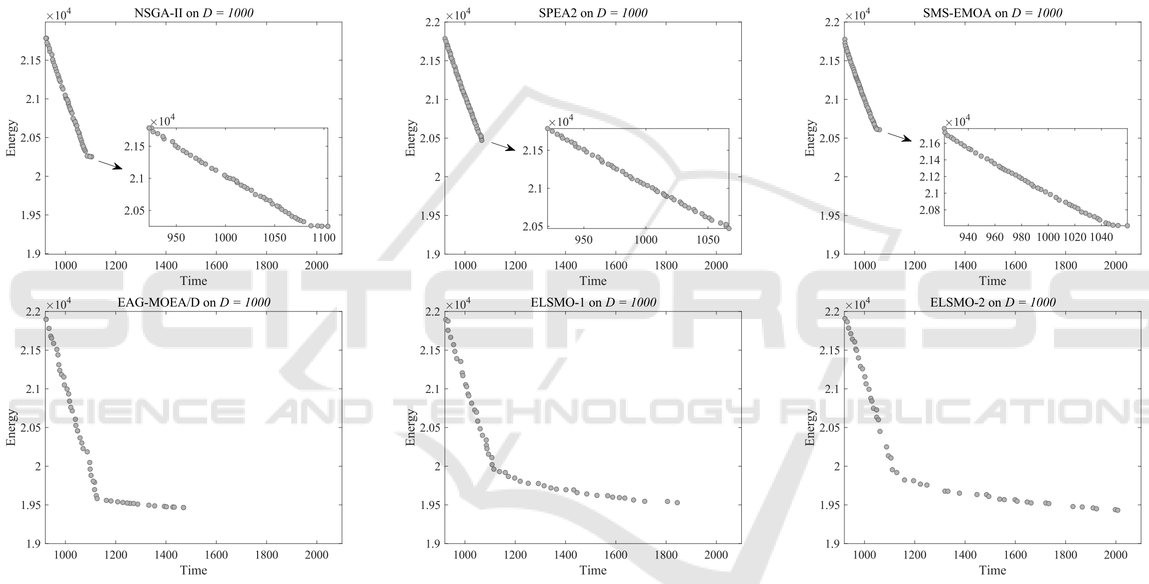


Figure 4: The non-dominated solution set with the medium HV value obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on the 1000-dimensional offloading problem.

is clear from the figures that ELSMO-2 can always get better performance between convergence and diversity no matter what the dimensional offloading problem.

5.3 Comparison with Other Offloading Schemes

It has been observed that ELSMO-2 can get the best non-dominated solution set in the Pareto front. The non-dominated solution set can give the decision-maker more choices. To further validate the performance of ELSMO-2, we compare the offloading gain between ELSMO-2 and EOS, COS and ROS based on the LOS. According to the different quality of service, the decision-maker may set different weights of w for the tradeoff between time and energy. If the decision-maker is sensitive to the time consumption, it may set a larger weight for the time consumption, or if the decision-maker focus is on energy performance, it may set a larger weight for the energy consumption.

Fig. 7, Fig. 8 and Fig. 9 presents the offloading gain of different offloading schemes under the different weights on 1000-, 5000-, 10000-dimensional offloading problems. It can be seen that offloading is always beneficial compared with the only local offloading scheme. And the proposed ELSMO-2 can always get the best offloading gain compared with other offloading schemes under different weights on different large-scale offloading problems. On the other hand,

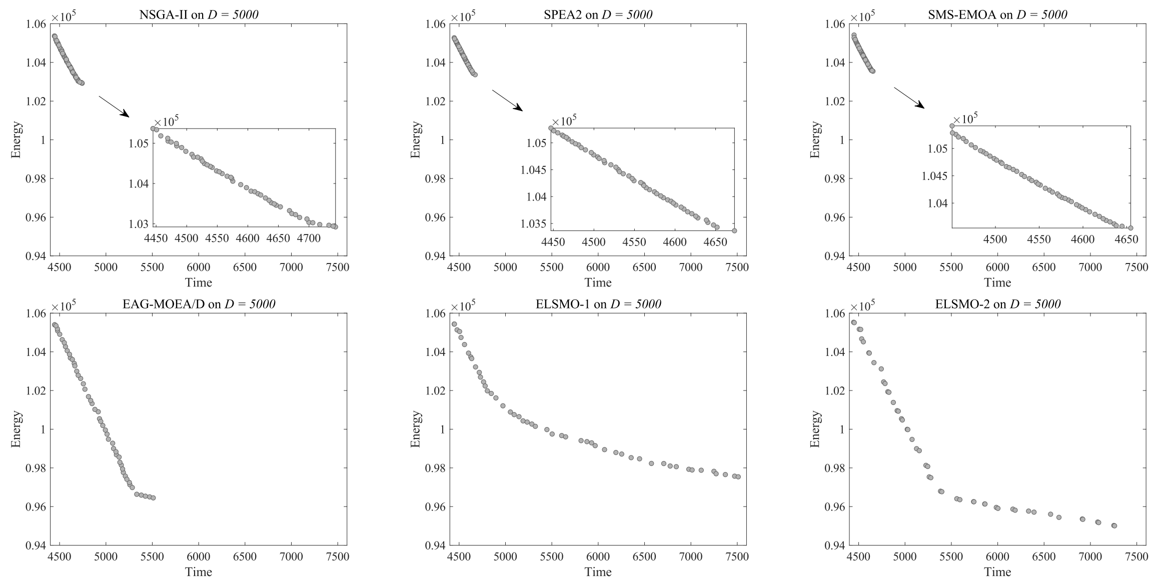


Figure 5: The non-dominated solution set with the medium HV value obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on the 5000-dimensional offloading problem.

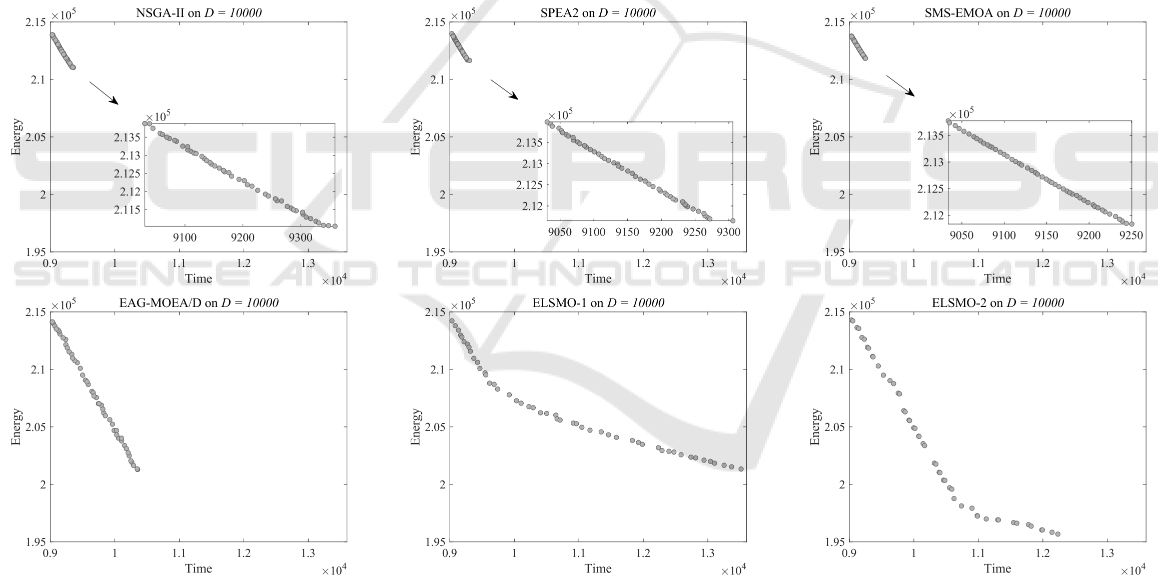


Figure 6: The non-dominated solution set with the medium HV value obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on the 10000-dimensional offloading problem.

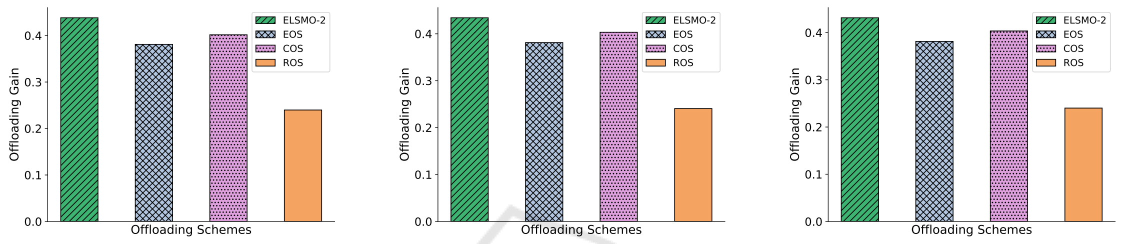
ELSMO-2 takes an obvious advantage over other offloading schemes when w is becoming larger, which means that ELSMO-2 can reduce the time delay more efficiently. What's more, compared with ROS, EOS and COS can obtain a better offloading gain, which also demonstrates that edge and cloud offloading can improve performance for both time and energy consumption.

6 CONCLUSION AND FUTURE WORK

In this paper, two evolutionary large-scale sparse multi-objective optimization (ELSMO) algorithms are proposed and compared for solving heterogeneous edge-cloud computation offloading problems. Considering the large-scale and sparsity properties of the multi-objective offloading model, the RBM is used to reduce the dimensionality and learn from the Pareto

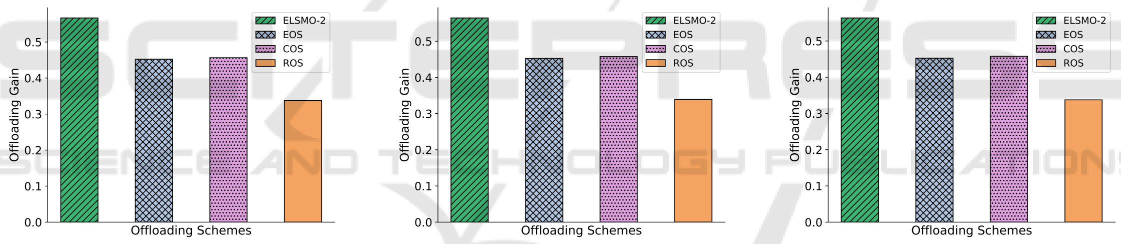
Table 3: The average HV values obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on offloading problems. Highlighted values corresponds to the best results according to the statistical tests.

Problem	D	NSGA-II	SPEA2	SMS-EMOA	EAG-MOEA/D	ELSMO-1	ELSMO-2
Offloading1	1000	2.7834e-1 (1.60e-3) –	2.7353e-1 (1.75e-3) –	2.6961e-1 (1.64e-3) –	2.9834e-1 (3.45e-4) +	2.9541e-1 (8.36e-4) –	2.9800e-1 (8.96e-5)
Offloading2	2000	2.6303e-1 (1.80e-3) –	2.5928e-1 (1.20e-3) –	2.5669e-1 (9.36e-4) –	2.9564e-1 (9.80e-4) –	2.6861e-1 (2.92e-2) –	2.9761e-1 (1.73e-4)
Offloading3	3000	2.5533e-1 (1.70e-3) –	2.5192e-1 (9.69e-4) –	2.5123e-1 (1.44e-3) –	2.8934e-1 (3.76e-3) –	2.8580e-1 (2.07e-3) –	2.9442e-1 (3.48e-4)
Offloading4	4000	2.5263e-1 (1.10e-3) –	2.4856e-1 (7.96e-4) –	2.4805e-1 (7.14e-4) –	2.8371e-1 (3.79e-3) –	2.7427e-1 (1.89e-2) –	2.9761e-1 (1.73e-4)
Offloading5	5000	2.5009e-1 (1.09e-3) –	2.4757e-1 (6.41e-4) –	2.4656e-1 (8.32e-4) –	2.8441e-1 (1.35e-3) –	2.7836e-1 (3.87e-3) –	2.9161e-1 (3.14e-4)
Offloading6	6000	2.4780e-1 (6.85e-4) –	2.4587e-1 (6.99e-4) –	2.4483e-1 (4.67e-4) –	2.8219e-1 (2.73e-3) –	2.7564e-1 (2.51e-3) –	2.9018e-1 (2.00e-4)
Offloading7	7000	2.4759e-1 (1.04e-3) –	2.4493e-1 (1.59e-4) –	2.4391e-1 (2.20e-4) –	2.7632e-1 (5.08e-3) –	2.6431e-1 (1.77e-2) –	2.8837e-1 (5.94e-4)
Offloading8	8000	2.4542e-1 (6.92e-4) –	2.4373e-1 (7.42e-4) –	2.4254-1 (6.27e-4) –	2.7476-1 (5.20e-3) –	2.7134e-1 (1.67e-3) –	2.8626e-1 (8.37e-4)
Offloading9	9000	2.4439e-1 (4.68e-4) –	2.4325e-1 (3.76e-4) –	2.4218-1 (7.11e-4) –	2.7607e-1 (3.52e-3) –	2.6904e-1 (2.81e-3) –	2.8521e-1 (3.74e-4)
Offloading10	10000	2.4434e-1 (3.23e-4) –	2.4267e-1 (5.62e-4) –	2.4212-1 (3.56e-4) –	2.7024e-1 (7.04e-4) –	2.7013e-1 (2.49e-3) –	2.8359e-1 (1.17e-3)
$+/-/\approx$		0/10/0	0/10/0	0/10/0	1/9/0	0/10/0	



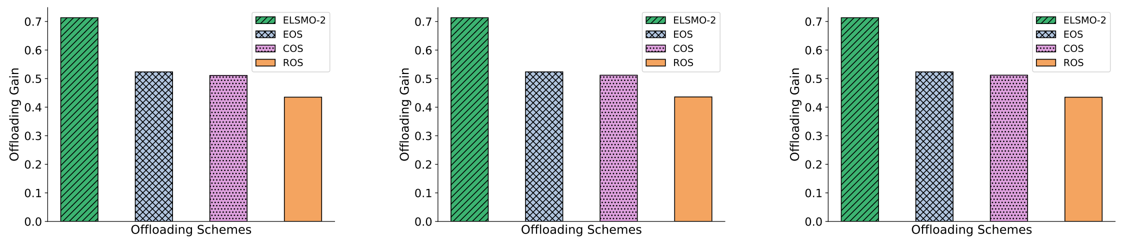
(a) $D = 1000$ (b) $D = 5000$ (c) $D = 10000$

Figure 7: Offloading gain of different offloading schemes for $w = 0.2$.



(a) $D = 1000$ (b) $D = 5000$ (c) $D = 10000$

Figure 8: Offloading gain of different offloading schemes for $w = 0.5$.



(a) $D = 1000$ (b) $D = 5000$ (c) $D = 10000$

Figure 9: Offloading gain of different offloading schemes for $w = 0.8$.

optimal subspace. While the contribution score is applied to select better decision variables to generate offspring. The proposed algorithms are compared with other MOEAs and offloading schemes to solve the test problems under different scales. The experimental results demonstrate the effectiveness and efficiency of the proposed algorithms.

In the future, some other efficient methods that can reduce the dimensionality will be considered, such as Principal Component Analysis (PCA) and

Autoencoder (AE). And also, the relationship between different decision variables will be investigated.

REFERENCES

Bi, S. and Zhang, Y. J. (2018). Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Transactions on Wireless Communications*, 17(6):4177–4190.

Cai, X., Li, Y., Fan, Z., and Zhang, Q. (2015). An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization. *Evolutionary Computation IEEE Transactions on*, 19(4):508–523.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Guo, F., Zhang, H., Ji, H., Li, X., and Leung, V. C. M. (2018). An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. *IEEE/ACM Transactions on Networking*, 26(6):2651–2664.

He, C., Li, L., Tian, Y., Zhang, X., Cheng, R., Jin, Y., and Yao, X. (2019). Accelerating large-scale multiobjective optimization via problem reformulation. *IEEE Transactions on Evolutionary Computation*, 23(6):949–961.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Hochstrate, N., Naujoks, B., and Emmerich, M. (2007). Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181:1653–1669.

Huang, L., Feng, X., Feng, A., Huang, Y., and ping Qian, L. (2018). Distributed deep learning-based offloading for mobile edge computing networks. *Mobile Networks and Applications*, pages 1–8.

Mach, P. and Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, 19(3):1628–1656.

Sahu, I. and Pandey, U. S. (2018). Mobile cloud computing: Issues and challenges. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 247–250.

Sheikh, I. and Das, O. (2018). Modeling the effect of parallel execution on multi-site computation offloading in mobile cloud computing. In *Computer Performance Engineering*, pages 219–234.

Tian, Y., Lu, C., Zhang, X., Tan, K. C., and Jin, Y. (2020a). Solving large-scale multiobjective optimization problems with sparse optimal solutions via unsupervised neural networks. *IEEE Transactions on Cybernetics*, pages 1–14.

Tian, Y., Zhang, X., Wang, C., and Jin, Y. (2020b). An evolutionary algorithm for large-scale sparse multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 24(2):380–393.

Wu, H., Knottenbelt, W. J., and Wolter, K. (2019). An efficient application partitioning algorithm in mobile environments. *IEEE Transactions on Parallel and Distributed Systems*, 30(7):1464–1480.

Xu, X., Liu, Q., Luo, Y., Peng, K., Zhang, X., Meng, S., and Qi, L. (2019). A computation offloading method over big data for iot-enabled cloud-edge computing. *Future Generation Computer Systems*, 95:522–533.

Yu, S., Chen, X., Yang, L., Wu, D., Bennis, M., and Zhang, J. (2020). Intelligent edge: Leveraging deep imitation learning for mobile edge computation offloading. *IEEE Wireless Communications*, 27(1):92–99.

Zhang, X., Tian, Y., Cheng, R., and Jin, Y. (2018). A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):97–112.

Zille, H. and Mostaghim, S. (2019). Linear search mechanism for multi- and many-objective optimisation. In *EMO*.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. volume 3242.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117–132.