



A MECS redistribution algorithm for SDN-enable MEC using response time and transmission overhead

Runze Wu¹ · Xiang Ao¹  · Bing Fan¹ · Hailin Hu¹

Accepted: 7 December 2020 / Published online: 4 January 2021
© The Author(s) 2021

Abstract

The software-defined networks-enable mobile edge computing (SDN-enable MEC) architecture, which integrates SDN and MEC technologies, realizes the flexibility and dynamic management of the underlying network resources by the MEC, reduces the distance between the access terminal and computing resources and network resources, and increases the terminal's access to resources. However, the static distribution relationship between MEC servers (MECSs) and controllers in the multi-controller architecture may result in unbalanced load distribution among the controllers, which would degrade network performance. In this paper, a multi-objective optimization MECS redistribution algorithm (MOSRA) is proposed to decrease the response time and overhead. A controller response time model and link transmit overhead model are introduced as basis of an evolutionary algorithm which is proposed to optimize MECS redistribution. The proposed algorithm aims to select an available sub-optimizes individual by using a strategy based coordination transformation from Pareto Front. That is, when the master controller of the MECS is redistributed, both of the network overhead of the MECS to the controller and the response time of the controller to the MECS processing request are optimized. Finally, the simulation results demonstrate that the MOSRA can solve the redistribution problem in different network load levels and different network sizes within the effective time, and has a lower control plane response time, while making the edge network plane transmission overhead lower, compared with other algorithms.

Keywords Redistribution · Response time · SDN-enabled MEC · Transmission overhead · Evolutionary algorithm · Coordinate transformation

1 Introduction

5G wireless infrastructure needs to support a large number of devices and complex applications. This will require the network to support high scalability, ultra-low latency, high throughput, and reliable data transmission. These critical delays and bandwidth requirements have prompted the European Telecommunications Standards Institute (ETSI) to propose multi-access edge computing as a viable solution to push services to the edge and closer to end users. Compared with high-performance servers on the cloud, MEC servers have limited computing resources. It is necessary to handle the heterogeneity of requests from

terminal devices, that is, to meet the requirements of different requests for computing power and delay, and to achieve load balancing between MEC servers. An available way to solve this problem is to introduce an MEC coordinator into the network.

In the SDN-enable MEC architecture, MECSs are deployed on the edge access network, reducing the distance between the access terminal and the computing resources and network resources, and increasing the terminal's access to resources [1]. The MEC coordinator can be integrated with the controller and the centralized controller has a global view of the network. Collecting information from the network, the MEC coordinator acts as many applications [2]. Those applications connect to various agents of the correct MEC server and makes corresponding decisions based on various factors such as server functions, current load and its location.

✉ Xiang Ao
axeesoul@163.com

¹ North China Electric Power University, BeiJing, China

However, there are some challenges in the distributed control structure of SDN-enabled MEC architecture, such as DDoS attacks [3–5] and load balancing [6–8]. In wireless networks, the peer-to-peer quality of service in the network needs to be optimized [9] and control plane scalability of distributed networks [10]. A portion of the controller's computing power is consumed, while other controllers are idle, which may cause the controller's response time to increase. The load of the controller is not balanced, and the transmission overhead in the network may also increase. The current research on the controller is mainly to reduce the response time of the control by redistributing the controller load in SDN network [11, 12].

1.1 Related works

The switches migration problem (SMP) is an NP-hard bin packing problem [13–15]. In other words, it is a challenge to effectively solve the switch redistribution problem in a valid time. Tao Hu et al. [16] defined the migration efficiency based on the changes in controller load and balance caused by the redistribution switch, and they proposed efficiency-aware switch migration (EASM) algorithm to achieve load balancing at a minimum cost. The redistribution cost of switches is inevitable, and the message exchange cost cannot be ignored in this process of redistribution [17]. Switches with different request rates will impose different burdens on the controller. During the process of switch redistribution, it is necessary to monitor the load state of the switch and controller, but frequent switch redistribution will cause more control overhead and other costs [18]. Qiang He et al. [19] collected data such as the request rate of switches at the edge of the plane network, and predicted the request rate of other switches by their algorithms in a short period of time, which greatly reduced the cost of network load detection. The redistribution of switches will change the distance between switches and controllers generally, which results in the communication cost change in the network [20]. Yang Xu et al. [21] proposed BalCon algorithm to measure the cost of switch redistribution from the routing computational load and load cost. Controller deployment [13, 22] and task scheduling [23] based on network traffic prediction.

Basing on the matching principle, Hu Tao et al. [24] proposed a bidirectional matching strategy (BMS) to examine the redistribution process from the perspective of switch and controller respectively, which can quickly solve the redistribution problem of switches. However, when the performance of the controller is excessive, some controllers may have a high load because of the defects of the matching mechanism, while the other controllers are under loading or even empty. To avoid the disadvantages caused by using matching algorithm alone, Tao Wang et al. [25]

added the coalitional game algorithm to the stable matching algorithm. Due to the introduction of the game mechanism, the under loading controllers can also participate in the switch redistribution. To reduce the complexity of migration, Chuan'an Wang et al. [20] proposed the switch migration-based decision-Making (SMDM) algorithm. According to a controller load diversity matrix, controllers are divided to outmigration controllers (OM_S) and immigration controllers (IM_S) in their paper. Their algorithm will reduce the search space, so it can solve the switch redistribution problem in a very short time when the network load is low. However, a controller can belong to both the OM_S and IM_S at the same time when the load difference of the controller is large, which will lead to the solution time of their algorithm is too long or even no solution. The more switches allowed to be redistributed, the higher the load balance of the network, and the higher the demand for network performance, the more switches need to be redistributed [26].

Multiple objectives need to be considered for dynamic switch redistribution in SDN. In this paper, we consider reducing controller response time and transmission overhead. Optimizing the average response time of the controller can guarantee the quality of service of the data plane, and reducing the PACKET_IN transmission overhead of the data plane can save more bandwidth resources. In the redistribution problem of SDN switch, it is necessary to design an effective algorithm so that the switches can be rapidly redistributed in effective time. In order to solve the multi-objective optimization problem, we use the decomposition based multi-objective evolutionary algorithm (MOEA/D) framework to decompose the target problem into multiple scalar sub problems. On the one hand, we wish to get a lower redistribution of switch controller response time. On the other hand, due to the way of communication controller and switches is in-band communication, we also hope that redistribution of the switches and the controller message exchange will not cause more costs, so that we can save more link bandwidth for the resources-limited edge network plane. In addition, the redistribution of switches can readjust the load between controllers and improve the utilization of network resources in the control plane.

1.2 Contribution

In this paper, we comprehensively consider response delay transmission overhead and propose a switch redistribution method based on delay and overhead optimization. By minimizing the average response delay of the control plane, the load balancing of the controller is realized, and the network performance of the control plane can be improved. On the other hand, by minimizing the average

transmission overhead to reduce the hop number between controller and switch, at the same time adjust the load of the controller and improve the network performance. In this paper, we proposed a switch redistribution algorithm based on delay overhead optimization to minimize response delay and transmission overhead.

In the existing researches, the response time of the controller mainly considers the number of switch requests processed by the controller, and does not consider the influence of the hop number between switch and controller. The controller response time model constructed, in this paper, can balance the load between the controllers, balance the distribution of the requests of the switch PACKET_IN packets processed by the controller, and to a certain extent, reduce the overhead on the links related to the controller. In the link transmission cost model, solving the minimum transmission cost reduces the number of node hop number between the controller and the switch, and balances the amount of requests processed between the controllers, making the transmission delay between the controller and the switch more lower, at the same time make the load of the controller more balanced, so that the response delay of the controller is further reduced. MOSRA that we designed is roughly divided into two steps. The decomposition based evolutionary algorithm decomposes the target problem into multiple scalar sub problems instead of solving it as a multi-objective optimization whole. And by using the co-evolutionary mechanism between multiple sub-problems, the difficulty of fitness allocation is reduced. Therefore, the computational complexity of the algorithm is relatively low. The optimal solution of each scalar sub problem is obtained on the Pareto Front, and finally a set of Pareto optimal solutions of the target problem is obtained. Wei Fang et al. [27] demonstrated that on a convex Pareto Front in evolutionary algorithms, individuals with greater target fitness based on coordinate transformation are better. Therefore, the coordinate transformation strategy is adopted to screen the individuals with the maximum population spacing as the target solution, after the coordinate transformation on Pareto Front.

In this paper, our contributions are as follows:

- We formulate the model of controller's average response time and the model of transmission overhead, respectively. After that, we defined the improvement degree function, and then consider the redistribution problem of the MECS as the minimization of response time of control plane and PACKET_IN transmission overhead.

- We design MOSRA in MOEA/D algorithm framework. In the update strategy of genetic operators, we add maximum allowable limit MECS redistribution in genetic operators to reduce the search space of the algorithm. It is able to solve the MECS redistribution problem in effective time, and it can converge to the suboptimal solution which has a small gap with the optimal solution.
- The simulation results show that the MOSRA algorithm proposed by us has good effects on the average response time and transmission overhead of the controllers compared with other comparison algorithms, and it still performs well even the network load is high.

The rest of this article is structured as follows. Section 2 introduces the system model and describes the research problem. Section 3 presents the MOSRA algorithm we proposed. Section 4 describes how do we set the simulation conditions and analyze the simulation results. Section 5 concludes this article and the last part is the future work.

2 System model

In this section, we will explain the construction of network model, controller average response time model, and the PACKET_IN transmission overhead model as well as the symbols and formulas used. We focus on how to formulate accurate mathematical models of network performance indicators, and then formulate the MECS redistribution as multi-objective optimization problem.

2.1 Network model

We present the SDN-enabled MEC as illustrated in Fig. 1. Adding SDN to the MEC architecture can achieve greater flexibility and dynamics in the network. SDN allows a global view of the underlying network, so flow control rules can be applied to implement complex service chain solutions [28]. These flow control rules hosted on MECSs are called Flow Table. It can be used to manage the network of interconnected distributed MEC servers. MEC Coordinator Northbound Application (MEC-CNA) can be deployed in the SDN controller, which can be programmed to process various situations. The MEC coordinator northbound application can control the flow of the underlying network through the SDN controller.

The control plane considered in this paper is a multi-controller structure. The SDN controllers and a part of the MECS are connected through a southbound physical link,

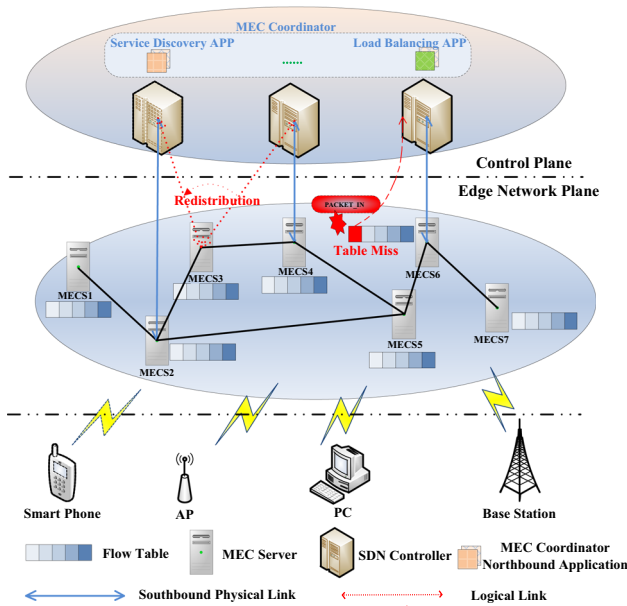


Fig. 1 SDN-enabled MEC architecture

while the other MECs and the SDN controller are logically connected. The request generated by those access devices will first be assigned to the closer MEC in the edge network plane. If there is no required application, the request will be routed to the MEC hosting the corresponding application. As shown in Fig. 1, there is no application requested by the access devices on MECS6, and the flow table on MECS6 generates Table-Miss. Then MECS6 will send PACKET_IN to the SDN controller to ask for route calculation. In the relevant literatures [16, 21, 29], PACKET_IN is the main component of the SDN controller load. So the load of the controller considered in this paper is all PACKET_IN sent by the MEC it serves.

In this paper, we denote a SDN-enabled MEC network as $G = (V, E)$, which consists of N MEC servers $\mathcal{S} = \{s_1, s_2, \dots, s_N\}$ and M controllers $\mathcal{C} = \{c_1, c_2, \dots, c_M\}$. Let $\mu = \{\mu_1, \dots, \mu_M\}$ denote the processing capacity of the controllers, which represents the number of PACKET_IN as the controller can handle per unit time. Specifically, in this paper, we do not consider the deployment of the SDN controllers and MEC servers in the network. Preserved a certain residual capacity for controllers, they can cope with the sudden increase of network traffic in a short period of time. We define the capacity usage limit factor of controllers, which indicates the maximum ratio of processing capacity that the controllers can use, and denote it as $\zeta = \{\zeta_1, \dots, \zeta_M\}$. The logical connection relationship between MECs and controller is defined as x_{ij} , where $x_{ij} = 1$ means that MEC i is logically distributed to controller j ,

and $x_{ij} = 0$ means that MEC i is not assigned to controller j . According to the Open Flow protocol, one switch can only be connected to a master controller at the same time. Therefore, we stipulate that one MEC can only be served by a SDN controller, so x_{ij} satisfies the condition that $\sum_j x_{ij} = 1, i \in \mathcal{S}, j \in \mathcal{C}$.

2.2 Controller response time model

In order to redistribute the load on the controller through delay optimization, in this section we will model the response time of the controller. Assumed that the request rate of PACKET_IN sent by the MECs follows the Poisson distribution with parameter λ_i [13], so the load of the controller is

$$\theta_j = \sum_i \lambda_i x_{ij} \tag{1}$$

One controller needs to process PACKET_IN for all MECs it serves, so it will generate a queue to be processed within the controller. We assume that the queue conforms to the M/G/1 queuing service model in queuing theory, and the request rate λ_i between MECs is independent of each other. The average sojourn time of PACKET_IN packet in controller j can be calculated as follow

$$\tau_j^s = \left(\frac{\theta_j}{2(\mu_j - \theta_j)} + 1 \right) \frac{1}{\mu_j} \tag{2}$$

where μ_j denotes the processing capacity of controller j .

To facilitate calculation, the response time of the controller that we considered consists of the sojourn time of the PACKET_IN packet in the controller and the transmission time between MECs. The average response time of controller j can be calculated as follow

$$\tau_j^r = \tau_j^s + \frac{\tau_0}{\sum_i x_{ij}} \cdot \sum_i d_{ij} x_{ij} \tag{3}$$

The previous term represents the average propagation time of all MECs served by the controller j , and the latter item represents the average transmission time between two adjacent nodes in the network. During the transmission process, the transmission delay between adjacent nodes does not differ much, so we treat it as a constant.

The average response time of all controllers in the control plane is

$$\bar{\tau}^r = \frac{1}{M} \sum_j \left(\frac{\theta_j}{\sum_j \theta_j} \cdot \tau_j^r \right) \tag{4}$$

Before the implementation of a MEC redistribution scheme, the average response time $\bar{\tau}_0^r$ of the control plane

was known in advance. In order to evaluate the improvement degree of an allocation scheme on the average response time, we formulate a normalized function to quantify the improvement degree. The closer the value of the function is to 0, the better effect of the distribution scheme on the improvement of delay is

$$\mathcal{G} = \exp\left(1 - \frac{\overline{\tau^r}}{\tau_0^r}\right) \tag{5}$$

2.3 Links transmission overhead model

The network’s dynamic changes in time and space as well as the demand for diversity lead to a high request rate of PACKET_IN, so it is necessary to evaluate the overhead of a large number of PACKET_IN during the transmission. The link transmission overhead model will be described below. The average transmission overhead of a PACKET_IN to controller j is calculated as follow.

$$\varepsilon_j = \frac{\sum_i v d_{ij} x_{ij} \lambda_i}{\theta_j} \tag{6}$$

where v denotes the overhead of a single-hop transmission of PACKET_IN, and d_{ij} is the hop number from MECS i to controller j . Then we can calculate the average overhead from MECSs to controllers across the network by Eq. (6).

$$\varepsilon = \frac{1}{N} \sum_i \left(\sum_j \varepsilon_j x_{ij} \right) \tag{7}$$

Before the implementation of a MECS redistribution scheme, we can calculate ε_0 , the average transmission overhead of the control plane. And we formulate a standardized function as Eq. (7). The closer the function value is to 0, the better the average transmission overhead is improved by this redistribution scheme.

$$\mathcal{M} = \exp\left(1 - \frac{\varepsilon}{\varepsilon_0}\right) \tag{8}$$

2.4 Problem description

In this paper, we formulate the average response time model and the transmission cost model to minimize the average response time and transmission overhead. Because there is a large dimensional difference between the values of the two objective functions, it should not to solve the multiple objects directly by aggregation. Fortunately, the MOEA/D class algorithm allows solving the objective functions after standardization. We establish improvement

assessment function for the average response time and average overhead in this paper, and design the MECSs redistribution algorithm. The mathematical model of the problem is expressed as P1.

P1:

$$\min_{x_{ij}} \mathcal{G} \tag{9}$$

$$\min_{x_{ij}} \mathcal{M} \tag{10}$$

$$s.t. \theta_j \leq \zeta_j \ominus \mu_j, \forall j \tag{11}$$

$$\sum_j x_{ij} = 1, \forall i \tag{12}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \tag{13}$$

$$\delta \leq \delta^{max} \tag{14}$$

In order to find the solution to the target problem, we set some constraints here, including controller processing capacity constraints, binary constraints and the maximum number of allowed redistribution MECSs.

In Eq. (10), $\zeta_j \in (0, 1)$ represents the limiting factor of the controller capacity, and reserves a certain processing capacity for the controller to deal with the burst flow in the network. The Eqs. (11), (12) are binary constraints of the MECS, ensuring that there is only one controller served for one MECS.

Equation (13) restricts the number of MECS redistributed in a redistribution scheme. Redistribute the logical relationship between the MECSs and the controllers, that is, the control logic of the MECS is transferred between the controllers. This process requires frequent communication with the MECS and controllers, which consume a certain amount of controller calculation and memory resources [24]. Therefore, in a redistribution scheme, if a large number of MECSs perform redistribution operations at the same time will cause the resource consumption of controllers in the entire network to increase sharply, which may increase the controller response time and reduce the reliability of the network. In general, the maximum allowable redistribution quantity is an important factor that cannot be ignored. Due to differences in controller performance and quantity, different networks can support different numbers of concurrent redistributions, so we set different maximum redistribution number for networks of different network sizes. The maximum number of MECSs allowed to redistributed in a redistribution scheme is

$$\delta^{max} = \lfloor \kappa N \rfloor \tag{15}$$

where $\kappa \in (0, 1)$ represents the limiting factor of the maximum redistribution quantity of MECs, which is determined by the network size. The function $\lfloor \cdot \rfloor$ means rounding down.

3 Multi-objective optimization MECs redistribution algorithm

In this section, we propose a multi-objective evolutionary algorithm to acquire the redistribution of MECs for the SDN-enabled MEC architecture.

3.1 MOEA/D algorithm

P1 will be solved in this section. As a mathematical model for multi-objective problems is constructed in this paper, considering that the SMP problem has been proved to be a NP-hard binary constraint backpack problem, we use a multi-objective evolutionary algorithm with good performance to solve multi-objective problems. The algorithm is divided into two parts. First, the optimal Pareto Front of the P1 problem is solved by MOEA/D. Then, the individual distance after coordinate transformation between the solutions on the Pareto Front is calculated, and we choose the individual with the smallest coordinate transformation distance as output. The chosen individual represents the logical connection relationship between the MECs and controllers.

In the multi-objective problem solving algorithm, compared with the weighted aggregation method, the Chebyshev aggregation method can better handle the non-convex situation of the Pareto Front. Based on the Chebyshev aggregation method in this document, the multi-objective problem in this paper is expressed as

$$\begin{aligned} \min F(x) &= \min(\mathcal{G}(x_{ij}), \mathcal{M}(x_{ij})) \\ &= \min(f_1(x), f_2(x)) \end{aligned} \tag{16}$$

In Eq. (15), f_1 and f_2 are the two objective functions in the multi-objective problem. So we can convert P1 to

$$\min F(x|\omega, z^*) = \max\{\omega_1(f_1(x) - z_1^*), \omega_2(f_2(x) - z_2^*)\} \tag{17}$$

s.t. (12) ~ (15)

In Eq. (16), $z^* = (z_1^*, z_2^*)$ is the reference point for the objective function value, and $z_i^* = \min\{f_i(x)\}$. There is a weight vector for any individual on Pareto Front to make the optimal solution of the Eq. (16), and the solution is also the Pareto optimal solution of the original problem [30].

A coordinate conversion strategy was used to find a relatively compromised solution between multiple objectives on the Pareto Front. The application of the coordinate transformation strategy to the model solving in this paper will be introduced below.

Suppose that there are two individuals p_1 and p_2 on Pareto Front. Considering individual p_1 as the origin of the coordinate system of the coordinate transformation, the objective function value of individual p_2 is adjusted to

$$f'_k(p_2) = \begin{cases} f_k(p_1) - f_k(p_2) & , \text{if } f_k(p_1) > f_k(p_2) \\ 0 & , \text{otherwise} \end{cases}, \quad k = 1, 2 \tag{18}$$

Individual p_2 in the reference system with individual p_1 as the origin, the inter-individual distance after coordinate transformation is

$$dis(f(p_1), f'(p_2)) = \sqrt{(f'_1(p_2))^2 + (f'_2(p_2))^2} \tag{19}$$

Then the distance between individuals p_1 in the population is

$$DIS_{f(p_1)} = \frac{1}{pop - 1} \sum_{e=1}^{pop-1} dis(f(p_1), f'(p_e)) \tag{20}$$

In Eq. (19), pop is the total population number. DIS reflects the quality of the individual p_1 corresponding solution.

In the coordinate transformation strategy, the fitness of individual p_e is normalized to

$$F_e = DIS_e - \frac{\min(dis_e)}{\max(dis_e)} - \min(dis_e) \tag{21}$$

3.2 MOSRA algorithm

Based on the MOEA/D algorithm, we design and implement the MOSRA algorithm. Different from the objective function weight aggregation method commonly used in the previous research of SDN load balancing algorithms, we use the MOEA/D algorithm to transform a multi-objective

problem into a set of single-objective scalar optimization problems. The pseudo code is shown in Algorithm 1.

Algorithm1 MOSRA

- 1: **Input:** controller processing capacity μ , capacity constraint factor ζ , PACKET_IN request rate distribution λ , iteration number g , population pop , number of weight’s neighbors T
- 2: **Output:** Redistribution of connection relationship x_{ij}
- 3: Initialize controller deployment, initial connection relationship x_{ij} ; $PF = \emptyset$, uniformly distributed set of weight vectors $\omega = \{\omega_1, \dots, \omega_r, \dots, \omega_{pop}\}$;
- 4: find T nearest weights for each weight vector, $B_r = \{B_1, \dots, B_r\}$
- 5: randomly generate pop feasible solutions set $pop = \{x_1, \dots, x_{pop}\}$
- 6: calculate the two objective function value by $f_1 = \{f_1(x_1), \dots, f_1(x_{pop})\}$ and $f_2 = \{f_2(x_1), \dots, f_2(x_{pop})\}$
- 7: initial reference point based on $z^* = (z_1^*, z_2^*)$, $z_i^* = \min\{f_i(x)\}$ and Eq. (16), (17)
- 8: While $g > 0$
- 9: Traversing the population pop
- 10: Randomly select two weight vectors in B_r , they are ω_1, ω_2 in ω , respectively
- 11: generate new feasible individual by cross and mutation, obtain y
- 12: Repair new individual with the constraint of Eq. (12), (15), and obtain y'
- 13: if $z_1^* < f_1'(y') \parallel z_2^* < f_2'(y')$
- 14: Update the $z^* = (z_1^*, z_2^*)$
- 15: if $F(y' | \omega, z^*) \leq F(x | \omega, z^*)$, set $x = y'$
- 16: Remove all solutions dominated by y' , and obtain PF
- 17: $g = g - 1$
- 18: Traversing the Pareto optimal solution PF
- 19: Calculate and based on Eq. (18), (19), (20)
- 20: Calculate and select the individual p corresponding to the maximum value F_c
- 21: Map the individual to x_{ij}

4 Performance evaluation

In this section, we will show the advantages and disadvantages of the proposed algorithm. The comparison algorithms are SMDM, SM, MORSA and SAS (static allocation schemes). Three environments are set up in the simulation, which are the scale-free network (SFN), SWITCH and OS3E network in Internet2 topology zoo.

For simplicity of calculation, we represent the processing power of the controller as the number of requests that can be processed per unit time.

4.1 Simulation setting

For the simulation we used Intel(R) Xeon(R) CPU E5-1620 v4@3.5GHz, 3.49GHz, 8 Cores, RAM 64G, running Matlab-R2016b. The population of MORSA algorithm is 100, and the number of iterations is 200. We performed a total of 100 times to take the average during the simulation (Table 1).

4.2 Results and analyses

The simulating results show that, if the number of migration constraints is set to 0.5, the algorithm proposed in this paper can calculate the result within the effective time. In the simulation, the maximum concurrent migration number factor κ is set to 0.5, that is, the number of MECS redistribution in each scheme does not exceed half of the number of MECSs. Figure 2 shows that under the condition of $\lambda = 18000$ packets/s, the final algorithm of this paper convergence of PF. The horizontal and vertical axes are the normalized values of the two objective functions, respectively. Specially, because we use discrete λ for simulation, the interval between request rates is 1000 packets/s, so the response time and overhead curves on the simulation diagram only showed a dozen points, and some images show a surge.

As shown in Figs. 3, 4 and 5, the comparison of the average response delay as the request rate increases of the controller in the network OS3E, SFN and SWITCH solved by the SMDM, SM and MORSA algorithm respectively. As shown in legend SAS, it can be seen from the Figs. 3, 4 and 5 that when the MECS and controller adopt a static allocation relationship, the average response time of controllers in the network increases dramatically with the request rate increase. All three dynamic redistribution algorithms can effectively suppress the growth of controller response time.

Table 1 Simulation parameters

Parameter	Value
OS3E	34 nodes
SFN	42 nodes
SWITCH	42 nodes
PACKET_IN Request rate λ	17000 ~ 29000 packets/s
Processing capacity of controller μ	180000 packets/s
Capacity constraint ζ	0.85 + 0.1*Rand(1)

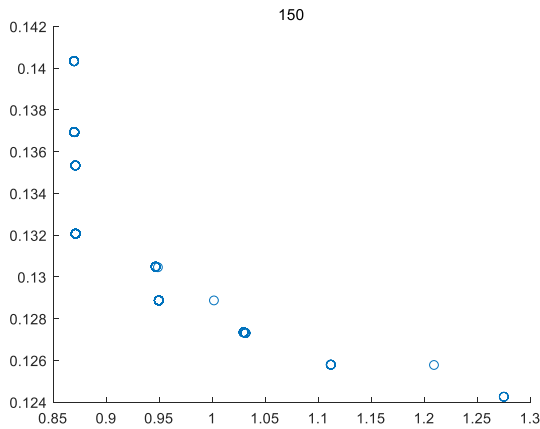


Fig. 2 Pareto Front distribution at $\lambda = 18000$ packets/s, generation = 150

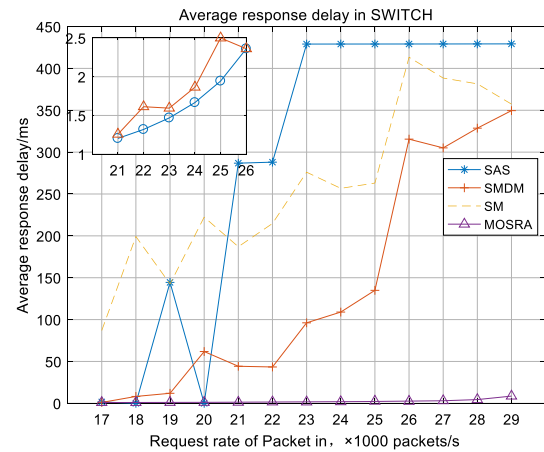


Fig. 5 Comparison of the average time in SWITCH

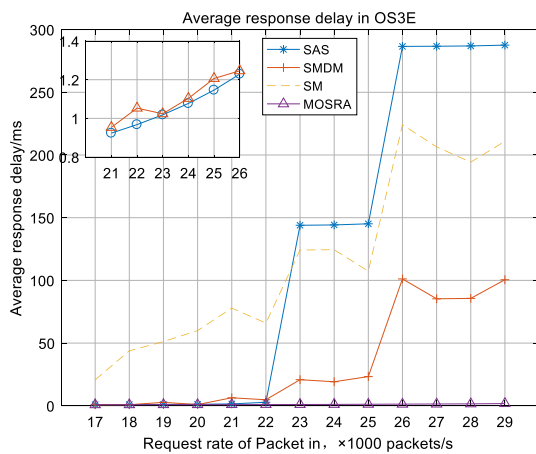


Fig. 3 Comparison of the average delay in OS3E

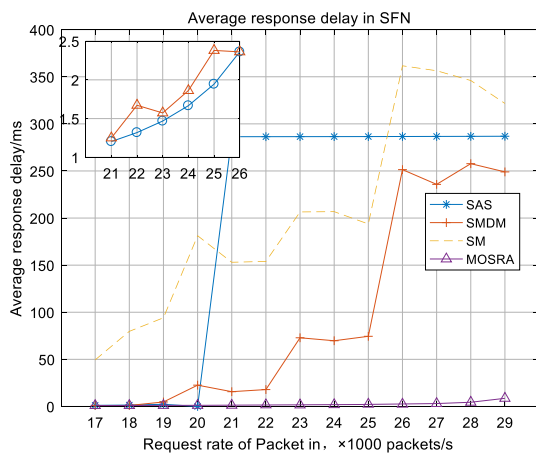


Fig. 4 Comparison of the average delay in SFN

For the SM algorithm, because the MECs and controllers use the *Galer-Shapley* bi-directional selection mechanism, the MECs has priority selection, and the MECs are preferentially assigned to the controller with

better processing performance, which means that the controller with larger processing capacity cannot reject the allocation. As the result, some high-performance controllers will have a larger load, but poor-performance controllers will not serve the MECs.

It can be seen that the average response delay of the SM first shows an upward trend, which is because the load actually taken by the controller increases. Then it shows a downward trend, which is because the number of controllers that actually participating in load sharing has increased. For example, when $\lambda = 23000$ packets/s, 5 controllers were sufficient to handle these loads while satisfying the controller not overload. Due to the bi-directional selection mechanism of Gale-Shapley, there remains 2 controllers were not serve the MECs, thus there is only 3 controllers participate in actually serving, which would result the average response time is at a high level in all three kinds of networks. When $\lambda = 24000$ packets/s or $\lambda = 25000$ packets/s, the 5 controllers no longer meet the processing requirement. The number of controller that actual sharing load reaches 6, which means the average load of the controller decreases, so the response time also decreases. The SMDM algorithm is essentially a greedy algorithm. It allocates a part of the heavier MECs from the heavier controllers to the lighter controllers, and repeats this until all controllers' load to be at a similar level. Therefore, the response time is at a low level when $\lambda < 26000$ packets/s, but rises sharply while $\lambda > 26000$ packets/s.

As shown in Figs. 6, 7 and 8, the comparison of the average overhead as the request rate increases in the network OS3E, SFN and SWITCH solved by the SMDM, SM and MORSA algorithm respectively. The SMDM algorithm multiplies the request rate and the hop number to build the controller's load model. The dimension of the hop number is much smaller than the request rate in the load

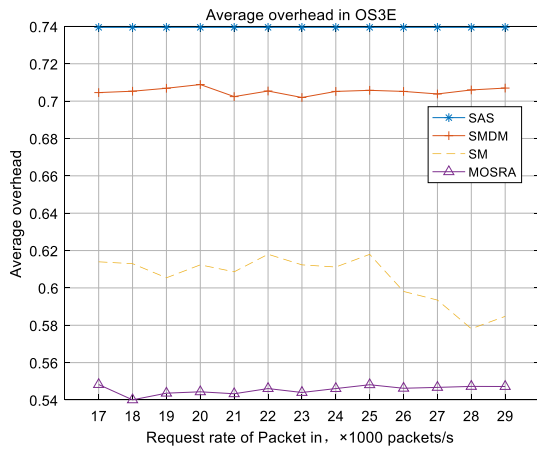


Fig. 6 Comparison of the average overhead in OS3E

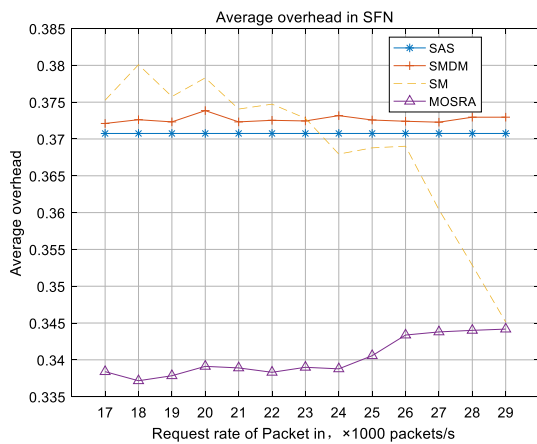


Fig. 7 Comparison of the average overhead in SFN

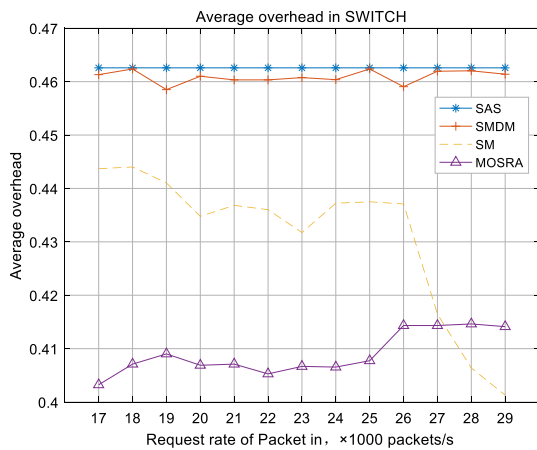


Fig. 8 Comparison of the average overhead in SWITCH

balancing problem. Therefore, the average cost cannot be guaranteed when using the greedy strategy. So the average cost curves shown in the graph are all at similar levels.

In the bi-directional selection process of the SM algorithm, the MECSs preferentially select the controller with

better processing performance, and the controller will reject the MECS with a longer distance when the controller load threshold is exceeded, so its average cost will fluctuate and fluctuate at that time. It is the same as the average response delay analysis of the SM algorithm. This is because the number of controllers that actually share the load increases. It can be seen in Fig. 6 that there is a small fluctuation when $\lambda < 26000 \text{ packets/s}$, and a significant decrease occurs when $\lambda > 26000 \text{ packets/s}$, as well as the average overhead began to rise when $\lambda > 28000 \text{ packets/s}$, which was caused by the increase of the average load shared by the controller.

Because only the request rate is considered in the construction of the load model, and the hop number is considered in the cost model, a multi-objective mathematical model is constructed, and the average delay and average overhead can be guaranteed when solved in MORSA. In the case of weights, MOEA/D is used to solve Pareto Front. Individual selection on Pareto Front is not determined by weights. Instead, it is selected based on the difference in the objective function value between individuals. Selecting the largest individual distance after coordinate transformation can better achieve a compromise solution between the two objective functions.

As shown in Figs. 9, 10 and 11, the comparison of the average response delay as the number of controller increases in OS3E, SFN and SWITCH solved by the SMDM, SM and MORSA algorithm respectively.

It is obvious that the network can reduce the average response delay and average overhead of the controller by increasing the number of controllers, but reducing the load and transmission overhead of the controller by adding controllers is not ideal. It can be seen in Figs. 9, 10 and 11 that when the network load approaches the processing capacity of the controller, increasing the controller can effectively reduce the average response delay. The controller number is less than 9 approximately. When the

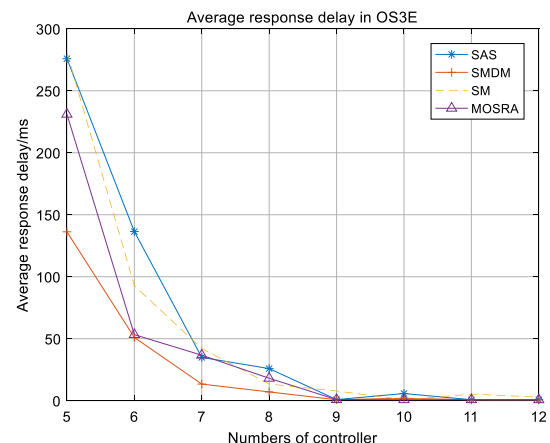


Fig. 9 Comparison of the average delay in OS3E

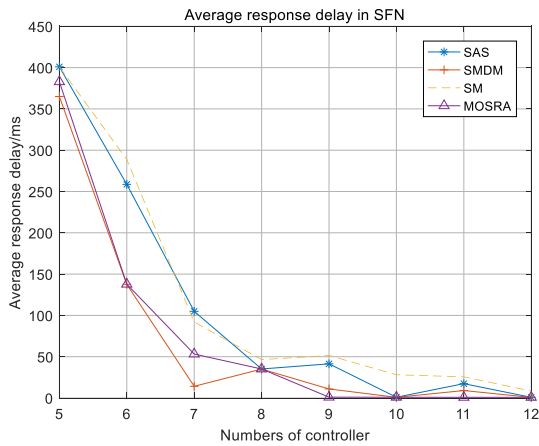


Fig. 10 Comparison of the average delay in SFN

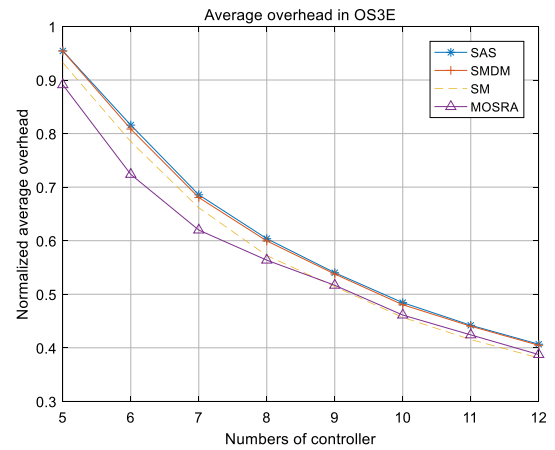


Fig. 12 Comparison of the average overhead in OS3E

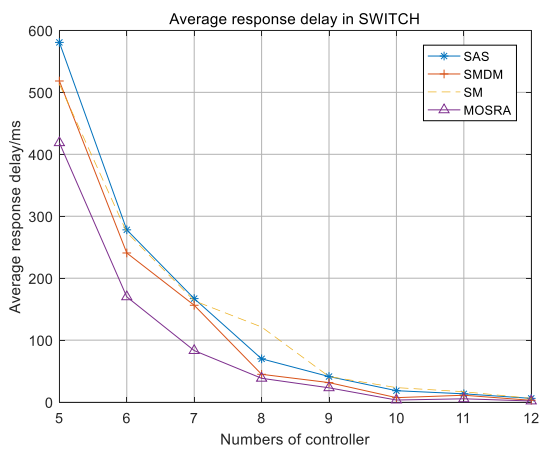


Fig. 11 Comparison of the average delay in SWITCH

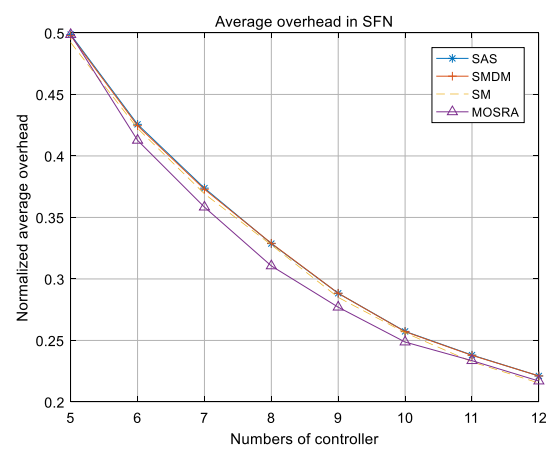


Fig. 13 Comparison of the average overhead in SFN

controller’s number reaches a certain number, the average response delay of the network is stably maintained within a certain range and does not decrease. This is because when reassigning, requests on a MECS cannot be distributed. After the controller exceeds a certain number, the average load of the controller will not decrease.

As shown in Figs. 12, 13 and 14, the comparison of the average overhead as the number of controller increases in OS3E, SFN and SWITCH solved by the SMDM, SM and MORSA algorithm respectively. Simply increasing the number of controllers in the network cannot effectively reduce the average overhead. From the Figs. 12, 13 and 14, it can be seen that the average overhead declining trend of all algorithms is almost the same as the static allocation scheme. In addition, it can be seen in the Fig. 14 that the average overhead of the MORSA algorithm in the three networks is at the bottom, indicating that its adaptability in the network is not bad.

Load balancing of controller is achieved through MECSs’ redistribution, so it is necessary to consider both the hop number and the load of controllers. Therefore, the solution of SMP depends on the network topology and

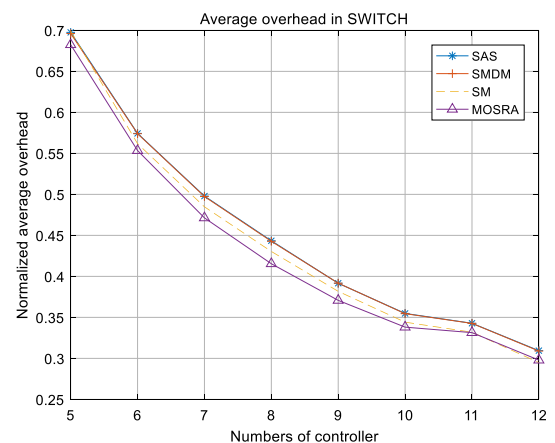


Fig. 14 Comparison of the average overhead in SWITCH

network load. The simulation results show that the SMP solution is closer to the hop number, when the MECS has a higher request rate.

Increasing the number of network controllers shortens the average distance from the MECS to the right controller

and reduces the average network overhead. Increasing the number of controllers also reduces the average load on the controller and reduces the average response delay of the network. But when increasing the number of controllers in the same network reaches a certain number, the average response delay and the average network overhead can no longer be reduced.

When the number of controllers is changed and deployed them in the network, the network topology also changes. Therefore, when the number of controllers in a network is large, the network performance is not necessarily better than when there are fewer.

5 Conclusion

In this paper, we introduce a MOEA algorithm to redistribute the master controller of MECS for improving network performance in SDN-enable MEC scenarios. The proposed algorithm uses MOEA to obtain Pareto optimal solution set and a coordinate transformation strategy to select an available solution from the set. This work aims at optimizing network performance by considering response time of controller and overhead of network. That is, when optimizing the MECSs redistribution plan, we can better reduce the response time and overhead, if we comprehensively considerate of the controller's computing ability and load, MECS request rate and the number of hops between MECS and the controller. Compared to the OS3E and SWITCH, the Scale-Free Network has serious uneven distribution of the connection status (degrees) between each node. It can be seen from the simulation results, in the SFN, increasing the number of controllers without considering the topological characteristics between the controller and MECS may even lead to a decrease in network performance. The network load level has an obvious influence on the response time of the controller, especially in the uneven network. The phenomenon of controller overload is more serious in the network. Compared with uneven network (SFN), the overhead of even network (OS3E) is higher relatively. Based on the test bed, we simulated the MOSRA and evaluated its performance in different topologies, rate of PACKET_IN and number of controllers, which demonstrated the feasibility and advantage of the proposed algorithm.

6 Future work

SDN-enabled MEC can achieve greater flexibility and dynamics in the network. Programmable hierarchical network will better support MEC task offloading and cloud-edge collaboration. For our future work, based on the

proposed MOSRA, we will investigate energy consumption at controller within complex network environment and focus on efficient control plane scaling algorithm design.

Acknowledgments The authors would like to send special thanks to the Professor LIANGRUI TANG for the wise suggestion in the research introduced in this study.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Funding This work is supported by the National Natural Science Foundation of China (Grant No. 51677065) and the Fundamental Research Funds for the Central Universities 2019QN-103.

References

1. Gu, X. H., Jin, L., Zhao, N., & Zhang, G. A. (2019). Energy-efficient computation offloading and transmit power allocation scheme for mobile edge Computing. *Mobile Information Systems*, 2019, 9.
2. Mavromatis, A., Colman-Meixner, C., Silva, A. P., Vasilakos, X., Nejabati, R., & Simeonidou, D. (2020). A software-defined IoT device management framework for edge and cloud computing. *IEEE Internet of Things Journal*, 7(3), 1718–1735.
3. Krishnan, P., Duttagupta, S., & Achuthan, K. (2019). SDNFV based threat monitoring and security framework for multi-access edge computing infrastructure. *Mobile Networks & Applications*, 24(6), 1896–1923.
4. Yan, Q., Yu, F. R., Gong, Q., & Li, J. (2016). Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey some research issues, and challenges. *IEEE Communications Surveys & Tutorials*, 18(1), 602–622. <https://doi.org/10.1109/COMST.2015.2487361>.
5. Linticum, D., As cloud use grows so will rate of DDoS attacks, Feb. 2013, [online] Available: <http://www.infoworld.com/d/cloud-computing/cloud-use-grows-so-will-rate-of-ddos-attacks-211876>.
6. Mao, Y. Y., Zhang, J., & Letaief, K. B. (2016). Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12), 3590–3605.
7. Liu, J. Q., Wan, J. F., Zeng, B., Wang, Q. R., Song, H. B., & Qiu, M. K. (2017). A scalable and quick-response software defined vehicular network assisted by mobile edge computing. *IEEE Communications Magazine*, 55(7), 94–100.
8. Cui, J., Lu, Q., Zhong, H., Tian, M., & Liu, L. (2018). A load-balancing mechanism for distributed SDN control plane using response time. *IEEE Transactions on Network and Service Management*, 15(4), 1197–1206. <https://doi.org/10.1109/TNSM.2018.2876369>.

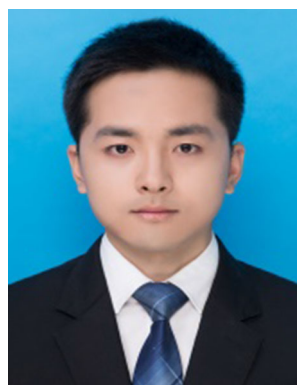
9. Gheisari, M., Alzubi, J., Zhang, X., et al. (2020). A new algorithm for optimization of quality of service in peer to peer wireless mesh networks. *Wireless Netw.* 26, 4965–4973.
10. Li, M., Wang, X., Tong, H., Liu T., Tian, Y. (2019) SPARC: Towards a scalable distributed control plane architecture for protocol-oblivious SDN networks. In: *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, Valencia, Spain, pp. 1-9, doi: <https://doi.org/10.1109/ICCCN.2019.8846931>.
11. Neghabi, A. A., Navimipour, N. J., Hosseinzadeh, M., & Rezaee, A. (2018). Load balancing mechanisms in the software defined networks: A systematic and comprehensive review of the literature. *IEEE Access*, 6, 14159–14178.
12. Aslan, M., & Matrawy, A. (2016). On the impact of network state collection on the performance of SDN applications. *IEEE Communications Letters*, 20(1), 5–8. <https://doi.org/10.1109/LCOMM.2015.2496955>.
13. Al-Tam, F., & Correia, N. (2019). Fractional switch migration in multi-controller software-defined networking. *Computer Networks*, 157, 1–10.
14. Li, G., Wang, X., & Zhang, Z. (2019). SDN-based load balancing scheme for multi-controller deployment. *IEEE Access*, 7, 39612–39622. <https://doi.org/10.1109/ACCESS.2019.2906683>.
15. Wang, Y., & You, S. (2018). An efficient route management framework for load balance and overhead reduction in SDN-based data center networks. *IEEE Transactions on Network and Service Management*, 15(4), 1422–1434. <https://doi.org/10.1109/TNSM.2018.2872054>.
16. Hu, T., Lan, J., Zhang, J., & Zhao, W. (2019). EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking. *Peer-to-Peer Networking and Applications*, 12(2), 452–464.
17. Cui, J., Lu, Q., Zhong, H., Tian, M., & Liu, L. (2018). A load-balancing mechanism for distributed SDN control plane using response time. *IEEE Transactions on Network and Service Management*, 15(4), 1197–1206.
18. Song, P., Liu, Y., Liu, T., & Qian, D. (2017). Flow stealer: Lightweight load balancing by stealing flows in distributed SDN controllers. *Science China-Information Sciences*. <https://doi.org/10.1007/s11432-016-0333-0>.
19. He, Q., Wang, X., & Huang, M. (2018). OpenFlow-based low-overhead and high-accuracy SDN measurement framework. *Transactions on Emerging Telecommunications Technologies*, 29(2), e3263.
20. Wang, C., Hu, B., Chen, S., Li, D., & Liu, B. (2017). A switch migration-based decision-making scheme for balancing load in SDN. *IEEE Access*, 5, 4537–4544.
21. Xu, Y., Cello, M., Wang, I. C., Walid, A., Wilfong, G., Wen, C. H. P., et al. (2019). Dynamic switch migration in distributed software-defined networks to achieve controller load balance. *IEEE Journal on Selected Areas in Communications*, 37(3), 515–529.
22. Yao, G., Bi, J., Li, Y., & Guo, L. (2014). On the capacitated controller placement problem in software defined networks. *IEEE Communications Letters*, 18(8), 1339–1342.
23. Huang, X., Bian, S., Shao, Z., & Xu, H. (2020). Predictive switch-controller association and control devolution for SDN systems. *IEEE/ACM Transactions on Networking*. <https://doi.org/10.1109/TNET.2020.3021787>.
24. Hu, T., Yi, P., Guo, Z. H., Lan, J. L., & Zhang, J. H. (2018). Bidirectional matching strategy for multi-controller deployment in distributed software defined networking. *IEEE Access*, 6, 14946–14953.
25. Wang, T., Liu, F., & Xu, H. (2017). An efficient online algorithm for dynamic SDN controller assignment in data center networks. *IEEE-Acm Transactions on Networking*, 25(5), 2788–2801.
26. Min, Z., Hua, Q., Zhao, J (2017) Dynamic Switch Migration Algorithm with Q-learning towards Scalable SDN Control Plane. In: *2017 9th International Conference on Wireless Communications and Signal Processing, International Conference on Wireless Communications and Signal Processing*.
27. Fang, W., Zhang, L., Yang, S., Sun, J., & Wu, X. (2019). A multiobjective evolutionary algorithm based on coordinate transformation. *IEEE Transactions on Cybernetics*, 49(7), 2732–2743.
28. Wang, J., Zhao, L., Liu, J., & Kato, N. (2019). Smart resource allocation for mobile edge computing: A deep reinforcement learning approach. *IEEE Transactions on Emerging Topics in Computing*. <https://doi.org/10.1109/TETC.2019.2902661>.
29. Li, G., Wang, X., & Zhang, Z. (2019). SDN-based load balancing scheme for multi-controller deployment. *IEEE Access*, 7, 39612–39622.
30. Zhang, Q. F., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6), 712–731.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Runze Wu received the Ph.D. degree in Communication and Information system from Beijing University of Posts and Telecommunications, China, in 2007, and the M.Tech. degree in Power System Automation in 2001 and the B.Tech. degree in Communication Engineering in 1998, from Dongbei Dianli University, Jilin China. She is currently an associate Professor with the Department of Electrical and Electronics Engineering School, North China Electric

Power University. Her current research interests include big data transaction, Physical Cyber System modeling, WSN and smart grid communications and information system planning.



Xiang Ao received the B.Sc. degree in communication engineering from North China Electric Power University in July 2018. He is currently pursuing his M.S. degree in electronics and communications engineering at North China Electric Power University. His research interests include Software Defined Networks, Mobile Edge Computation.



Bing Fan received the Ph.D. degree in power system and automation from North China Electric Power University (NCEPU), Beijing, P. R. China, in 2015. He is currently an Associate Professor with the State Key Laboratory of Alternate Electrical Power System with Renewable Energy Sources of NCEPU. His current research interests include wireless networks and optical networks for power system.



Hailin Hu received the B.Sc. degree in Telecommunications Engineering from North China Electric Power University in July 2016. He is currently pursuing his Ph.D. degree in Electrical Engineering of North China Electric Power University. Now his research interests cover edge intelligence, resource allocation.