PAPER

Research on the Switch Migration Strategy Based on Global Optimization

Xiao'an BAO[†], **Shifan ZHOU**[†], **Biao WU**[†a)], **Xiaomei TU**[†], **Yuting JIN**[†], **Qingqi ZHANG**[†], *and* **Na ZHANG**[†], ***Nonmembers***

SUMMARY With the popularization of software defined networks, switch migration as an important network management strategy has attracted increasing attention. Most existing switch migration strategies only consider local conditions and simple load thresholds, without fully considering the overall optimization and dynamics of the network. Therefore, this article proposes a switch migration algorithm based on global optimization. This algorithm adds a load prediction module to the migration model, determines the migration controller, and uses an improved whale optimization algorithm to determine the target controller and its surrounding controller set. Based on the load status of the controller and the traffic priority of the switch to be migrated, the optimal migration switch set is determined. The experimental results show that compared to existing schemes, the algorithm proposed in this paper improves the average flow processing efficiency by 15% to 40%, reduces switch migration times, and enhances the security of the controller.

key words: *software-defined network, whale optimization algorithm, load balancing, load prediction, switch migrations*

1. Introduction

With the continuous expansion of network scale and the increase in complexity, traditional network management methods can no longer meet the requirements for network performance, reliability, and flexibility [1]. Software-Defined Networking (SDN), as an emerging network architecture, has brought revolutionary changes to network management by separating network control and data plane [2].

As the core component of SDN architecture, SDN controller is responsible for centralized control and management of the entire network [3]. In order to improve the scalability, fault tolerance and performance of the network, SDN controllers are often deployed in the form of clusters. However, with the expansion of SDN network scale and diversification of application scenarios, the existing SDN controller cluster architecture faces some challenges, one of which is the migration of switches [4].

In the SDN controller cluster, switches may need to be migrated to different controller nodes, which is caused by controller node failure, load balancing needs, or network topology changes [5]. However, the switch migration process may lead to network interruption, traffic loss, performance degradation and other problems, affecting the normal operation of the network [6].

Manuscript received December 11, 2023.
Manuscript revised March 1, 2024.
Manuscript publicized March 25, 2024.
[†]The authors are with Zhejiang Sci-Tech University, China.
a) E-mail: biaowuzg@zstu.edu.cn
DOI: 10.1587/transinf.2023EDP7263

Therefore, it is of great significance to study the methods and strategies of switch migration in the SDN controller cluster for improving the availability, reliability and flexibility of the network. This research aims to explore an effective switch migration scheme to reduce network interruption time, minimize traffic loss and optimize network performance. The overview and analysis of existing migration mechanisms and algorithms can provide guidance for the design and implementation of switch migration in SDN controller clusters.

The main contributions of this paper are as follows:

1) In order to reduce the cost of switch migration, this paper proposes a switch migration algorithm based on global optimization (SMGO). Through the improved whale optimization algorithm, the most suitable controller is found according to the cost of migration path, controller load, and the number of idle controllers around the controller. The algorithm in this paper can ensure that the load on the control plane is well balanced after migration, reduce the migration cost, and ensure that important traffic is prioritized.
2) In order to reduce frequent migration and avoid unnecessary migration operations, this paper introduces the load prediction module into the migration model. The load forecasting module predicts the future network load and traffic patterns by analyzing historical data and trends. When the network load is predicted to change, the action module will make corresponding adjustments to avoid network congestion and performance problems.

2. Correlation Studies

An effective way to solve the performance and scalability constraints in traditional SDN implementations is to use distributed controllers. In order to improve the performance of the control plane in SDN, several aspects need to be considered. On the one hand, the performance can be improved by maximizing the performance of the controller and transferring part of the work to the forwarding device. On the other hand, a group of distributed controller nodes can be used to realize the logically centralized control plane and solve the consistency problem of global view and state, thus improving the scalability. However, when a large amount of uneven traffic reaches the distributed controller, this method may lead to the problem of load imbalance between con-

trollers [7].

Cheng et al. [8] proposed a game decision-making mechanism, which is used to migrate the switch from an overloaded controller to an idle controller while maximizing resource utilization. However, there is a lack of game triggering mechanism, and the whole process will generate additional network overhead. In the distributed nearest migration algorithm [9], the controller with the closest physical location is selected as the target controller for easy operation, but this method is likely to bring new load imbalance. In the maximum resource utilization migration algorithm (MUMA) [10], when load imbalance occurs, the controller randomly selects a switch for migration. However, the algorithm does not consider load balancing, and new overloads may occur after migration. Liu et al. [11] proposed a multi controller deployment algorithm based on delay and load, but it did not consider the future load situation, which may cause a large cost when the controller is migrated in the next step after migration. Gao et al. [12] proposed a multi controller dynamic load balancing mechanism. When the controller is overloaded, the super controller is used to balance the migration effect and the migration cost, and formulate an appropriate switch migration strategy. However, it takes a long time to calculate, and for different network architectures, the optimization effect varies greatly.

Through investigation at home and abroad, it is found that the control plane load imbalance problem is generally solved by switch migration strategy, but the existing switch migration strategy is often based on static planning or heuristic algorithm, lacking flexibility and adaptability, and can not adapt well to the rapidly changing network environment and traffic demand.

In order to solve the above problems, this paper uses load forecasting algorithm to improve the migration method. The algorithm can determine the trigger time of migration, and select appropriate immigration and emigration controllers to reduce unnecessary migration. If it is predicted that the controller that needs to be relocated will also be overloaded at the next moment, then make a comprehensive judgment based on the load of the surrounding idle controllers, and re-select the controller that needs to be relocated to reduce the migration frequency in the future. In addition, by using switch collection migration, you can reduce migration times and improve migration efficiency. At the same time, traffic is prioritized to ensure that important traffic can be prioritized.

3. Problem Description and Migration Model Modeling

The research background of this paper is SDN under distributed architecture. In SDN, the logical central controller (decision-maker) manages the data plane by sending data packets with processing policies to the switch. Because the traffic is unstable and unbalanced within a certain period of time, when the switch traffic in the controller domain suddenly increases, the response of the controller to the request will be affected, resulting in overload of the controller.

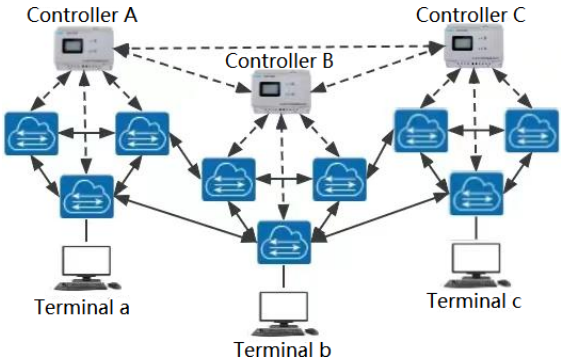


Fig. 1 SDN multi-controller architecture.

3.1 Problem Description

The common SDN controller cluster architecture is shown in Fig. 1. When a controller gathers a large number of requests and the controller is in an overload state, according to the definition of the OpenFlow protocol, if the main controller of the switch is in an overload or inefficient state, the switch can be migrated and another controller can be selected from the control plane as the new main controller [13].

3.2 Symbols and Definitions

The entire network architecture is represented by graph $G = (N, S)$, where N and S represent the set of nodes and links, respectively. The controller set is represented by C and the switch set by E. $T = [x_{ij}]_{M\times N}$ is the connection matrix of all elements, where M and N represent the number of controllers and switches, and x_{ip} represents the mapping relationship between switch E_i and controller C_p, as shown in Formula (1).

$$x_{ij} = \begin{cases} 1, & Switch\ E_i\ is\ connected\ to\ controller\ C_j \\ 0, & Switch\ E_i\ is\ not\ connected\ to\ controller\ C_j \end{cases} \tag{1}$$

The load value of the controller is defined as the number of switches Packet_In in its domain received by the controller, as shown in Formula (2).

$$LC_p = \sum_{i=1}^{m} P_{E_i C_p} \tag{2}$$

Where, LC_p represents the load value of the controller, and $P_{E_i C_p}$ represents the number of Packet_In received by the controller C_p from the switch E_i.

The controller C_p status is defined as α_p, and the value of α_p is shown in Formula (3).

$$\alpha_p = \begin{cases} 0, & LC_p < LC_{threshold} \\ 1, & LC_p \geq LC_{threshold} \end{cases} \tag{3}$$

Where, $LC_{threshold}$ is the threshold value of controller overload.

Table 1 Traffic priority division

Type	*Pri*	*Priority*
Background service	4	low
Interactive business	3	↓
Real time-stream	2	↓
Information control management	1	high

The standard deviation of controller load is used to express the load balance factor, as shown in Formula (4).

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(LC_i - \overline{LC})^2} \tag{4}$$

After the switch is migrated, the new load balancing factor is shown in Formula (5).

$$\sigma^* = \sqrt{\frac{1}{N}\left(\sum_{i=1,i\neq j}^{N}\left(LC_i{}^* - \overline{LC}^*\right)^2 + \left(LC_j{}^* - \overline{LC}^*\right)^2\right)} \tag{5}$$

Where $LC_i{}^* = LC_i - L_{E_iC_j}$, $LC_j{}^* = LC_j - L_{E_iC_j}$, $\overline{LC}^*$ is the new average load and $L_{E_iC_j}$ is the number of Packet_In received by the controller C_j from the switch E_i.

Link congestion rate, using the throughput and link bandwidth capacity The ratio is expressed.

$$q_{link} = \frac{B}{C} \tag{6}$$

Where q_{link} represents the congestion rate of the link and B represents the throughput of the link. Gets the number of bytes accepted and sent by the switch port and the time by sending a $PORT\_STATS\_REQUEST$ message to the switch. The throughput of the link is indicated by Formula (7).

$$B = \frac{T_2 - T_1}{t_2 - t_1} + \frac{R_2 - R_1}{t_2 - t_1} \tag{7}$$

Where T and R represent the bytes sent and received respectively, and t means unified Calculate the time. The congestion rate can be calculated by Formula (6) and (7).

3.3 Traffic Category Classification

In the migration process, not only the migration cost and load balance, but also the type of traffic needs to be considered, and important traffic needs to have a higher migration priority. To achieve this goal, this paper uses SDN combined with deep packet inspection (DPI) [14] to identify traffic. The traffic priority division scheme is shown in Table 1. Traffic is divided into four categories according to business types to ensure that important traffic can be prioritized.

4. A Globally Optimized Switch Migration Model

The migration model in this paper is shown in the figure. It involves four modules in total: load statistics module, traffic prediction module, migration target selection module and action management module. The corresponding functions of each module are as follows.

1) Load statistics module. This module is mainly responsible for counting the load of each controller in the controller cluster, judging whether the traffic in the controller exceeds the threshold, and determining the controller to be migrated.
2) Flow prediction module. This module receives the load statistics from the load statistics module and the controller to be migrated. The prediction model predicts the load situation of the controller to be migrated at the next moment according to the traffic history information to determine whether migration is still needed to avoid frequent migration.
3) Target selection module. This module is the core module of this paper, which is mainly responsible for finding the target controller and its surrounding controller clusters that can make the migration behavior of the entire controller cluster reach the global optimization.
4) Migration implementation module. This module receives the migration controller from the traffic prediction module and the migration controller from the target selection module, and is responsible for sending specific migration strategies to the data layer switch and migration, so as to achieve load balancing of the controller cluster.

This model divides the migration process into four modules. Each module is divided into work and cooperation, which improves the work efficiency to a certain extent. On the basis of the common functions of load detection and migration selection, the flow prediction function is introduced to avoid the waste of resources caused by unnecessary migration. The target selection module benefits from traffic prediction, which makes the controller target selected more reasonable and the whole network system more stable. The migration model is shown in Fig. 2.

4.1 Traffic Prediction Module

According to the current research, switches are mostly migrated when the controller is overloaded. However, in actual research, there may be situations where the load is too high for a short time, but the controller can handle and recover to normal working status immediately. In this case, the switch migration will cost more than waiting for the controller to complete the processing. Therefore, this paper introduces a load prediction model to avoid unnecessary migration and reduce resource waste.

Auto-Regressive Integrated Moving Average Model (ARIMA) is a common time series analysis method [15], which can be used to predict future traffic. The following steps are used for traffic prediction with ARIMA model:

Step 1: Collect data, collect historical traffic data, including traffic values at each time point.

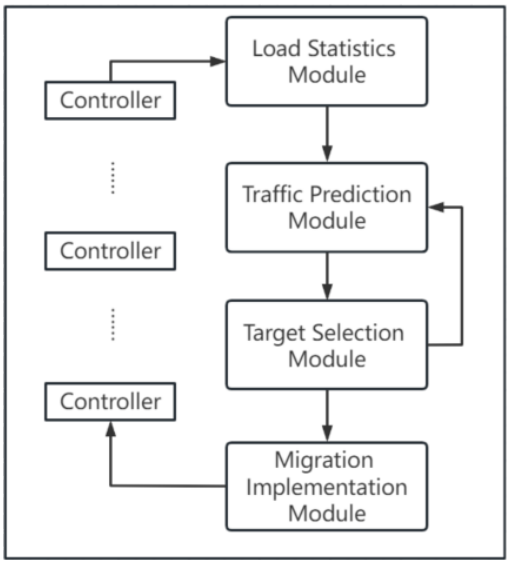


Fig. 2 Switch migration model.

Step 2: Data preprocessing, preprocessing data, including deleting outliers, checking data stability, etc. ARIMA model requires that the time series is stable, that is, the mean and variance do not change with time. If the data is unstable, it needs to be stabilized, such as difference operation.

Step 3: Model selection, select the appropriate ARIMA model according to the autocorrelation and partial autocorrelation graph of the data. Autocorrelation plots can be used to determine moving average components, and partial autocorrelation plots can be used to determine autoregressive components. Automated methods, such as grid search, can be used to determine the best ARIMA model.

Step 4: Model estimation, Based on the selected ARIMA model, the maximum likelihood estimation (MLE) is used to estimate the model parameters. This will result in an optimal set of model parameters.

Step 5: Model test, use the estimated model parameters to fit the historical data, and calculate the residual (the difference between the observation value and the model prediction value). The residual sequence is tested to ensure that it meets the requirements of randomness and stability.

Step 6: Model prediction: use the estimated ARIMA model to predict the future traffic of the controller.

The flow of algorithm 1 is as follows: after the load statistics module completes the statistics and preliminary evaluation, the traffic prediction module will receive the load information from the traffic statistics module to preliminary judge the overload controller and the target controller. The prediction model is built to predict the load at the next time. If the overload controller is still overloaded at the next time, select the controller as the exit controller and send the migration trigger signal $Move$.

4.2 Target Selection Module

This module is the core of the entire migration model. This

Algorithm. 1 Controller flow prediction algorithm

1) Load_information from Load_Statistics Module
2) Overload Controller C_1,Switch Collection λ_E
3) Training of the ARIMA model
4) $\longrightarrow$Predict the load situation of C_1 at the next moment, LC_1'
5) While $LC_1' > LC_{threshold}$
6) $C_1 \longrightarrow$ Overload; Move;
7) Target Controller C_2, Controller Collection λ_{C_2} from Target_selection_Module
8) if $LC_2' + L\lambda_E < LC_{threshold}$
9) Target $= C_2$;
10) else
11) Target $= C_{new} \in \lambda_{C_2}$;
12) Move_Start;
13) Update controller $< C_1, C_2 >$

paper proposes a global optimal migration algorithm based on the improved whale optimization algorithm to reduce the migration cost and ensure the performance of the controller.

4.2.1 Global Optimal Target Controller

The original Whale Optimization Algorithm (WOA) steps are roughly as follows:

1) Initialize individual position and velocity: The initial position $x(t)$ and velocity $v(t)$ are randomly generated for each whale.
2) Calculate individual fitness: calculate the fitness value of each whale according to the objective function of the problem.
3) Select the current optimal solution: according to the individual fitness, select the current optimal solution p.
4) Select the global optimal solution: select the global optimal solution g according to the fitness size of all individuals.
5) Update speed and position: update the velocity and position of each whale according to the following formula:

$$v(t+1) = A * (p - x(t)) + C * (g - x(t)) \quad (8)$$

$$x(t+1) = x(t) + v(t+1) \quad (9)$$

Where $v(t+1)$ represents the speed of the $t+1$ time step; $v(t)$ represents the speed of the t time step; A is the degree of freedom factor used to regulate the individual speed; p is the individual optimal solution; $x(t)$ is the position of the t time step; C is the amplitude scale factor, which is used to adjust the amplitude of the global optimal solution; g is the global optimal solution.

Because the traditional whale optimization algorithm randomly generates the initial position of the whale at the beginning, this randomness will lead to slow convergence of the algorithm, and there may be premature convergence problems, that is, it is easy to fall into the local optimal solution and cannot find the global optimal solution [16].

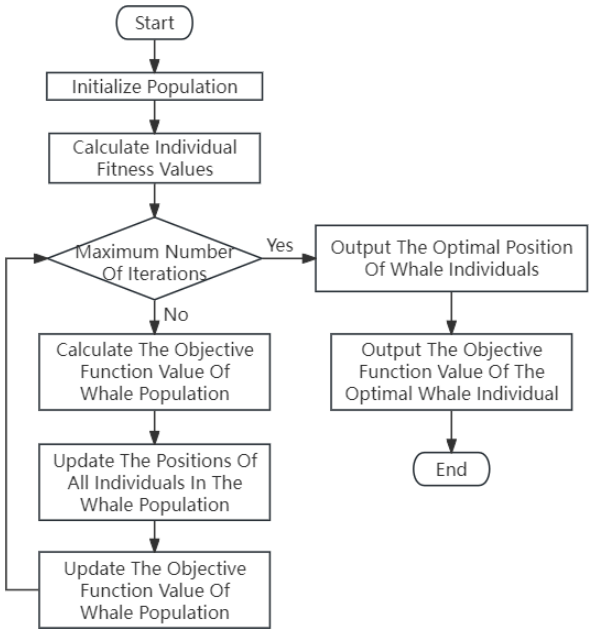


Fig. 3 AWOA algorithm flow.

So this paper proposes Adaptive Whale Optimization Algorithm (AWOA) based on adaptive inertia weight. In the process of position update, inertia weight is introduced. By adjusting the size of inertia weight, we can balance the global search and local search, and improve the performance of the optimization algorithm [17]. The updating strategy of inertia weight adopted in this paper is linear decreasing, which makes the inertia weight gradually reduce from the maximum value to the minimum value, thus increasing the global search ability of the algorithm. A larger inertia weight can accelerate the exploration ability of the algorithm, while a smaller inertia weight can enhance the local search ability of the algorithm. By gradually reducing the inertia weight, the algorithm can explore the solution space more widely at the initial stage of the search, and search the optimal solution more finely at the later stage of the search. The formula of inertia weight introduced on the original basis is as follows:

$$v(t+1) = w * v(t) + A * (p - x(t)) + C * (g - x(t)) \tag{10}$$

$$w = w\_max - (w\_max - w\_min) * (t-1)/T \tag{11}$$

Among them, w is the inertia weight, which is used to control the influence of the current speed, that is, balance the influence of the individual optimal solution and the global optimal solution on the speed; w_max is the initial maximum inertia weight; w_min is the minimum inertia weight; t indicates the number of current iterations; T represents the maximum number of iterations.

In the search and predation phase of AWOA algorithm, each whale individual is regarded as a specific solution, and the whale individual is evaluated according to the results of each iteration. When the maximum number of iterations is reached, the optimal whale individual in the current maximum number of iterations is regarded as the global optimal solution of the algorithm.

4.2.2 For the Selection of the Migration Node

At the beginning of the migration action, the switch nodes connected to the migration controller need to be added to the node set T_{Node} to be selected according to the migration priority.

Calculate the flow LC_{in} that the target controller can accept. LC_{in} is defined as the difference between the target controller load and the average load of the control plane controller, as shown in Formula (12).

$$LC_{in} = \left| LC_{target} - \frac{1}{n}\sum_{i=1}^{n} LC_i \right| \tag{12}$$

Where, LC_{target} is the target controller load.

The node selection problem is actually a dynamic programming problem. The optimization goal is to select the node with the highest priority under the limit of the total capacity LC_{in}.

1) Problem analysis

In the first i switch nodes, select several nodes to join the set T_{Node} to be migrated.

Status: In the first i switch nodes, select several nodes to join the set T_{Node} to be migrated with the remaining load space of lc to obtain the maximum priority.

Decision: whether to select the i switch node to join MigNode{ }.

2) The state transfer equation

Let it represent the maximum priority occupied by the migration collection T_{Node} with a load capacity of lc after joining the first i switch nodes

$$\mathrm{F(i,j)} = \max \begin{cases} \mathrm{F(i-1,lc - Load[i] + S\_Pri[i])}, & \text{select the ith node} \\ \mathrm{F(i-1,lc)} & , \textit{not select the ith node} \end{cases} \tag{13}$$

Where $F(0, j) = 0$, $Load[i]$ represents the load of the i node, and $S\_Pri[i]$ indicates the migration priority of the i node.

4.3 Migration Implementation Module

In this module, complete the switch migration operation, update the connection matrix $\Omega = [x_{ij}]_{M\times N}$ of the network elements, and change the role of the controller. First, through the overload controller and target controller determined by the prediction module, the overload controller selects the migration switching unit to be migrated, and sends the trigger migration signal to the target controller C_{target}. After receiving the signal, C_{target} will respond to the migration start signal and start the migration action.

Algorithm. 2 is based on a global optimal migration algorithm with an improved whale optimization algorithm

1) Initialize the position and velocity of the initial population ⟶ idp, idv
2) Initialize the global optimal position ⟶ gp_{best}
3) The global optimal fitness values were calculated ⟶ gf_{best}
4) iteration = 0, maxIterations = T
5) while iteration < T and not convergenceCriterion
6) for each i in population
7) update idp, idv
8) adjustPositionToBounds
9) evaluate individualFitness ⟶ idf
10) if idf > idf_{best}
11) idp_{best} = idp
12) idf_{best} = idf
13) if idf > gbf_{best}
14) gp_{best} = idp
15) gf_{best} = idf
16) iteration = iteration + 1
17) return gp_{best}

The detailed process of Algorithm 2 is as follows. First, the load prediction module confirms whether the overload controller needs to be migrated by analyzing historical data and trends. Then, the target selection module preliminarily selects the globally optimal target controller and its surrounding idle controller cluster through the continuous convergence iteration of the improved whale optimization algorithm, and finally determines the target controller by the prediction module. Finally, the action module sends the migration start signal to the overload controller to start the migration operation.

5. Analysis of Experimental Results

5.1 Algorithm Performance Test

In order to test the optimization performance of the adaptive inertia weight based whale optimization algorithm (AWOA) and other swarm intelligence optimization algorithms proposed in this paper, the benchmark function [18] is used for simulation testing, and the results are compared and analyzed.

In this paper, the classical benchmark function is used as the fitness function to compare the convergence accuracy and convergence speed of different optimization algorithms. Where, $F_1(x) \sim F_3(x)$ is a single peak benchmark function, and $F_4(x) \sim F_6(x)$ is a multi peak benchmark function. The experimental parameters of Particle Swarm Optimization (PSO) and WOA algorithms are set as follows: population size is 30, particle dimension is 30, and the maximum number of iterations is 500. After a large number of experiments, the value range of inertia weight in AWOA algorithm is set to 0.01 to 0.4, and other parameters are consistent with WOA algorithm.

The test results are shown in Fig. 4, where the convergence curve represents the optimal convergence value of the current iteration number. It can be seen from Fig. 4 (a) to Fig. 4 (f) that AWOA algorithm has a faster convergence rate in the iterative process than PSO and WOA algorithm. This is because the adaptive inertia weight strategy improved in this paper is introduced. The inertia weight changes adaptively with the population state, avoiding invalid iterations, increasing population diversity, and further improving the convergence speed and accuracy. It can be seen from Fig. 4 (b), Fig. 4 (c) and Fig. 4 (f) that the convergence curve of the AWOA algorithm is not displayed at the end of the iteration, which proves that the AWOA algorithm has converged to the optimal value at this time, and no redundant iterative search is required. However, other algorithms will fall into the local optimal value in the optimization process, and the convergence curve gradually becomes stable, so that they cannot jump out of the local optimal domain, hinder the global optimization efficiency, and lead to a significant decline in the overall optimization performance.

In addition, this article conducted 1000 independent localization optimization experiments in the same simulation environment. The experimental results are shown in Table 2. Although the AWOA algorithm has a slightly higher computational complexity than PSO, its optimization localization performance is better. After introducing an adaptive inertia weight strategy on the basis of the traditional WOA algorithm, the average consumption time of the AWOA algorithm is reduced by about 40%, while the average optimization positioning error is reduced by 28%, indicating better optimization performance.

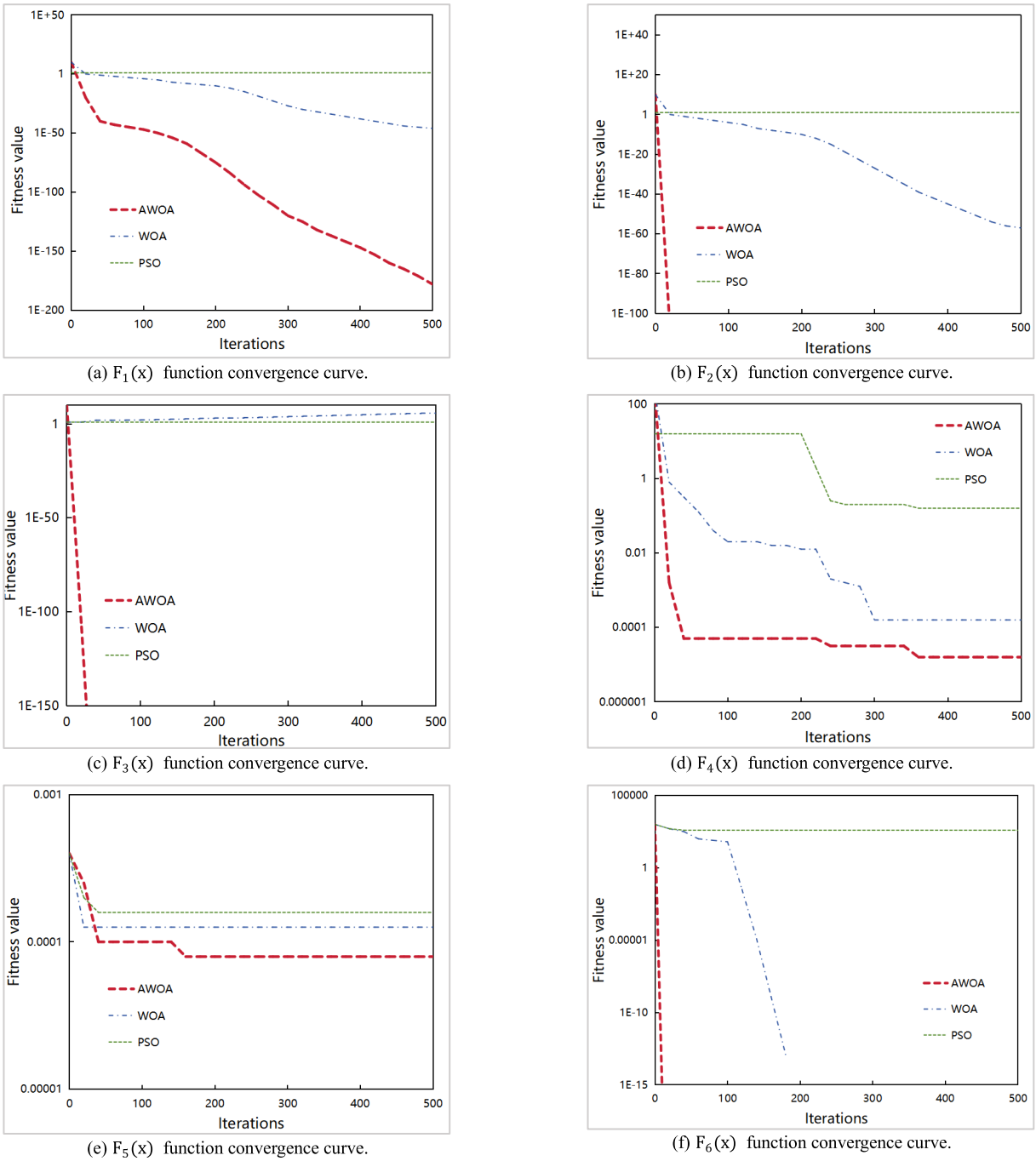
5.2 Experimental Environment

This experiment uses OpenDaylight to realize the control and management of SDN. Set multiple SDN controller instances on different virtual machines, and enable them to communicate with each other by configuring network connections between controllers. Connect the SDN switch to the controller cluster by specifying the IP address or domain name of the controller on the switch [19]. Use the Mininet tool to simulate and deploy the SDN topology, and use the Ipref tool to simulate the flow to test the performance of the controller. The topology structure adopted in this paper is: 5 controllers $C_0 \sim C_4$, 15 switches $E_0 \sim E_{14}$, which are randomly distributed. The experimental topology is shown in the Fig. 5.

Experimental parameters. Considering that the average diameter of a typical real network topology is about $log(n)$ [20], the maximum number of public nodes in the experiment is set to $2log(n)$, and the channel delay from the controller to the switch is 0.5ms. Each group of laboratory runs 20 times, and takes the average value to ensure unbiased.

5.3 Controller Load

This paper defines the controller load ratio as the ratio of the



(a) $F_1(x)$ function convergence curve.

(b) $F_2(x)$ function convergence curve.

(c) $F_3(x)$ function convergence curve.

(d) $F_4(x)$ function convergence curve.

(e) $F_5(x)$ function convergence curve.

(f) $F_6(x)$ function convergence curve.

Fig. 4 Convergence curve of benchmark test function.

Table 2 Algorithm average consumption time and average positioning error

Algorithm name	*Average consumption time/s*	*Average positioning error/m*
PSO	0.045	146.58
WOA	0.086	153.73
AWOA	0.052	110.69

actual controller receiving *Packet_In* request data packets to the controller capacity, as shown in the Formula (14).

$$\sigma = \frac{\mathrm{LC}}{\mathrm{C_m}} \tag{14}$$

This paper compares the maximum, middle and minimum load proportions in the controller cluster. Figure 6 and Fig. 7 clearly show the comparison of the load share of the controller using the SMGO algorithm mentioned in this article and not using any load balancing algorithm under the

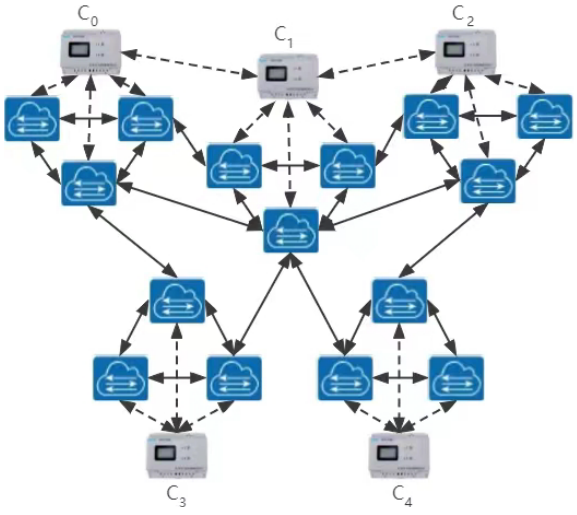


Fig. 5 Experimental topology.

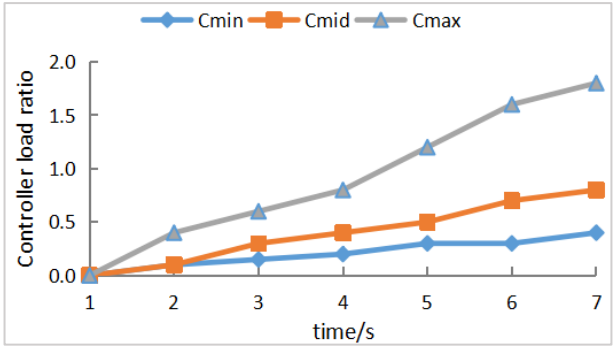


Fig. 6 Controller load distribution under static distribution.

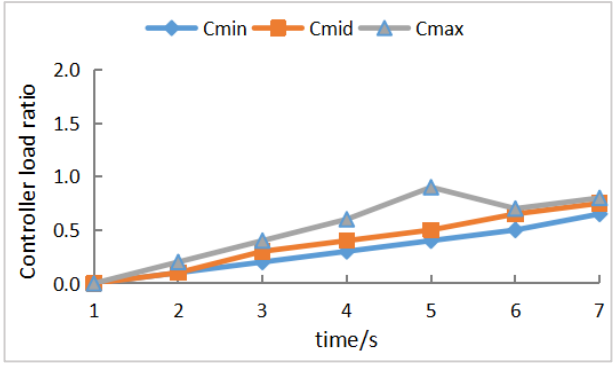


Fig. 7 Controller load distribution under SMGO.

same traffic transmission situation.

Figure 6 shows the load distribution of the controller after the network runs for a period of time without load balancing. Figure 7 shows the load distribution of the controller after SMGO algorithm is added. It can be seen from Fig. 7 that after implementing the SMGO algorithm, the number of loads exceeding the controller capacity is greatly reduced, and the network stability is better. In the case of static allocation, no measures are taken after the controller load exceeds

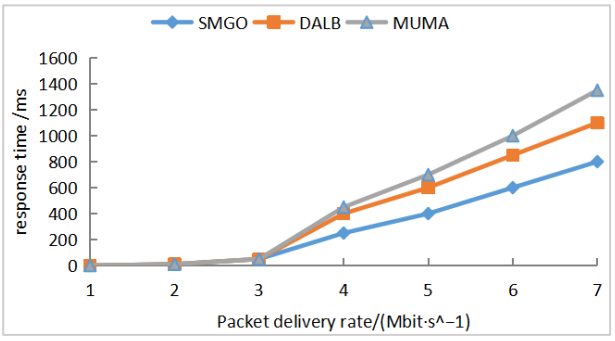


Fig. 8 Response time comparison.

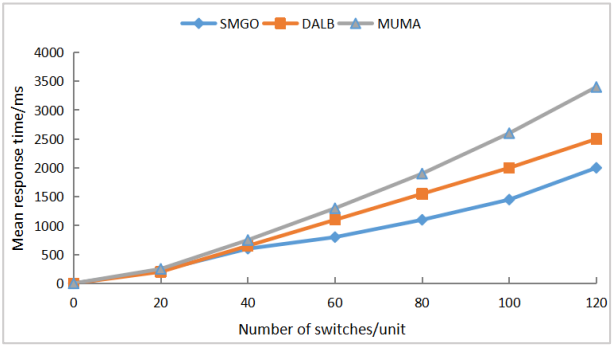


Fig. 9 Relations of network size and response times.

the threshold, and the resource allocation on the entire link is very unbalanced.

5.4 Response Time

The load balancing of different algorithms is measured by testing the response time of the controller. The controller load balancing dynamic adaptive algorithm (DALB) [21] and the maximum resource utilization migration algorithm (MUMA) are used for comparison.

The response time comparison is shown in Fig. 8. It can be seen from Fig. 8 that the controller response time of the SMGO algorithm proposed in this paper is shorter than that of the MUMA and DALB algorithms, and its performance is better than that of the MUMA and DALB algorithms. Figure 9 shows that the average response time of the controller changes with the increase of the network size when the transmission rate is unchanged. Experiments show that SMGO algorithm has shorter controller response time and better performance than the other two load balancing algorithms.

5.5 System Stability

The stability of the system is related to the number of migrations, because during the migration process, some switches may not be able to handle new connections in time, resulting in migration interruption and other problems [22]. At the current scale, the migration frequency of SMGO algorithm is significantly less than that of DALB and MUMA algo-

Table 3 Average number of migrations at different network sizes

Number of switches / unit	*MUMA / times*	*DALB / times*	*SMGO / times*
15	2	2	1
30	6	4	1
45	8	6	2
60	10	8	3

rithm. It can be seen from Table 3 that SMGO will consider the migration cost of the controller at the next moment every time the load is balanced. As the switch size continues to increase, the gap shows an obvious trend of expanding.

6. Conclusion

This paper studies the switch migration strategy under the distributed software definition network architecture, and proposes a switch migration strategy based on global optimization. The strategy first finds the target controller and the surrounding controller set based on the improved whale optimization algorithm. On this basis, the migration target is recalculated through the load prediction algorithm to ensure that this migration has the smallest impact on the network stability at the next moment. The simulation results show that AWOA algorithm has higher precision and faster convergence speed than PSO, WOA and other algorithms; When applied to the scenario of global optimal controller target location, the positioning performance of the algorithm in this paper is significantly better than that of the compared algorithm, and it is more suitable for switch migration in the SDN controller cluster environment.

References

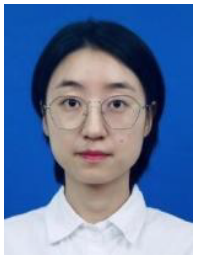
[1] S. Zhang and F. Zou, "Review of software-defined web research," Application Research of Computers, vol.30, no.8, pp.2246–2251, 2013.
[2] C.-K. Hang, Y. Cui, H.-Y. Tang, and J.-P. Wu, "State-of-the-Art Survey on Software-Defined Networking (SDN)," Journal of Software, vol.26, no.1, pp.62–81, 2015. DOI: 10.13328/j.cnki.jos.004701
[3] S. Wang, J. Li, Y. Zhang, et al., "The SDN architecture and security study," Telecommunications Science, vol.29, no.3, pp.117–122, 2013.
[4] X. Wang, "Multi-balancing deployment and switch migration strategy in SDN," Hebei University, 2021. DOI: 10.27103/d.cnki.ghebu.2021.000942
[5] F. Meng, "Study on load balancing based on SDN multicontroller," Anhui University, 2023. DOI: 10.26917/d.cnki.ganhu.2022.000488
[6] Y. Li, "Switch migration and real-time routing updates in software-defined networks," Tianjin Normal University, 2021. DOI: 10.27363/d.cnki.gtsfu.2021.000314
[7] W. Lee, "Research on Distributed Traffic Load Equilibrium Technology based on SDN," Southwest University of Science and Technology, 2023. DOI: 10.27415/d.cnki.gxngc.2023.000117
[8] G. Cheng, H. Chen, H. Hu, and J. Lan, "Dynamicswitch migration towards a scalable SDN control plane," International Book Titleof Communication Systems, vol.29, no.9, pp.1482–1499, 2016.
[9] S. Zhang, J. Lan, P. Sun, and Y. Jiang, "Online load balancing for distributed control plane in software-defined data center network," IEEE Access, vol.6, pp.18184–18191, 2018.
[10] A.A. Neghabi, N. Jafari Navimipour, M. Hosseinzadeh, and A. Rezaee, "Load balancingmechanisms in the software defined networks:a systematic and comprehensive review of theliterature," IEEE Access, vol.6, pp.14159–14178, 2018.
[11] X. Liu, B. Zhao, X. Fang, etc., "A study on SDN multicontroller deployment strategy based on MCDDL algorithm," Journal of Hubei Institute, vol.40, no.1, pp.50–57, 2022. DOI: 10.13501/j.cnki.42-1908/n.2022.03.009
[12] D. Gao, "Study on load balancing strategy based on SDN controller and traffic scheduling," Anhui University, 2023. DOI: 10.26917/d.cnki.ganhu. 2022.001551
[13] N. Zhang, X. Chen, and Y. Yang, "Research on Network Structure and Its Key Technology based on OpenFlow," Network Application Branch of China Computer Users Association, Proc. 26th Annual Conference of New Technology and Network Application 2022, [Publisher unknown], 2022:5. DOI: 10.26914/c.cnkihy.2022.049276
[14] A. Santos Da Silva, C.C. Machado, R.V. Bisol, L.Z. Granville, and A. Schaeffer-Filho, "Identification and selection of flow features for accurate traffic classification in SDN," Network Computing and Applications (NCA), IEEE, pp.134–141, 2015.
[15] Z. Li, Y. Hu, T. Hu, and P. Wei, "Dynamic SDN controller association mechanism based on flow characteristics," IEEE Access, vol.7, pp.92661–92671, 2019.
[16] A.A. Qaffas, S. Kamal, F. Sayeed, P. Dutta, S. Joshi, and I. Alhassan, "Adaptive population-based multi-objective optimization in SDN controllers for cost optimization," Physical Communication, vol.58, p.102006, 2023.
[17] K. Sridevi and M.A. Saifulla, "LBABC: Distributed controller load balancing using artificial bee colony optimization in an SDN," Peer-to-Peer Networking and Applications, vol.16, no.2, pp.947–957, 2023.
[18] S. Mirjalili and A. Lewis, "The whale optimization algorithm," Advances in Engineering Software, vol.95, pp.51–67, 2016.
[19] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of graph neural networks for network modeling and optimization in SDN," Proc. 2019 ACM Symposium on SDN Research. New York: ACM Press, pp.140–151, 2019.
[20] V. Huang, G. Chen, X. Zuo, A.Y. Zomaya, N. Sohrabi, Z. Tari, and Q. Fu, "Request Dispatching Over Distributed SDN Control Plane: A Multiagent Approach," IEEE Trans. Cybern., pp.1–14, 2023.
[21] G. Cheng, H. Chen, Z. Wang, and S. Chen, "DHA: distributed decisions on the switch migration toward a scalable SDN control plane," IFIP Networking Conference (IFIP Networking). IEEE, pp.1–9, 2015.
[22] Y. Zhang and M. Chen, "Performance evaluation ofSoftware-Defined Network (SDN) controllers using Dijkstra's algorithm," Wireless Networks, vol.28, no.8, pp.3787–3800, 2022.

Xiao'an Bao received the B.S. from Zhejiang University (China) in 1998, and M.S. from China West Normal University in 2004. He was an Associate Professor at Zhejiang Sci-Tech University, China, from 2007 to 2012. Since November of 2012, he has been a Professor at Zhejiang Sci-Tech University, China. His main research interests include software engineering and computer vision, and pattern recognition. First-Author is the author of over X technical publications, proceedings, editorials and books. He has published dozens of papers in journals such as Software Journal, Computer Research and Development, International Journal of Multimedia and Ubiquitous Engineering, and International Journal of Advancements in Computing Technology.

Shifan Zhou is currently studying at Zhejiang Sci-Tech University, pursuing master's degree in Computer Science.

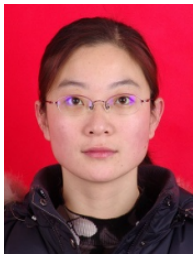
Biao Wu was born in 1989. He received the B.S. from Hubei University of Technology in 2013 and the M.E. from Zhejiang Sci-Tech University in 2016 both in the People's Republic of China. He received the Ph.D. from Yamaguchi University, Japan in 2021. Since April of 2021, he has been a Lecturer at Zhejiang Sci-Tech University. His main research interests include computer vision, deep learning and program net theory.

Xiaomei Tu was born in 1995, M.S. candidate. She from Zhejiang Sci-Tech University. Her research interests include video image processing and object detection.



Yuting Jin was born in 1994. Master. She from Zhejiang Sci-Tech University. Her research interests include computer vision, deep learning, program analysis and intelligent software testing.

Qingqi Zhang was born in 1996. He received the B.E. from Heilongjiang University, China, in 2019 and the M.E. from Zhejiang Sci-Tech University, China, in 2022. He is currently a Ph.D. candidate in the Graduate School of East Asian Studies, Yamaguchi University, Japan. His main research interests include computer vision and pattern recognition.



Na Zhang was born in 1977. She from Fenghua City, Zhejiang Province, is a professor with a master's degree. She from Zhejiang University (China), mainly engages in research on computer vision and intelligent information processing.