ChinaRxiv

AI translation · View original & related papers at
chinarxiv.org/items/chinaxiv-202205.00075

Research on SDN Multi-Controller Load Balancing Mechanism Based on Improved Genetic Algorithm (Postprint)

Authors: Xu Aixin, Sun Shimin, Wang Xiaofan, Xu Guowei, Meiyu Wang

Date: 2022-05-10T00:00:00+00:00

Abstract

To address the multi-controller load imbalance problem in software-defined networks, a master controller reselection model based on non-cooperative game theory for load shedding is proposed. First, dynamic thresholds are utilized to identify overloaded controllers. Second, a priority-based switch migration decision mechanism is adopted. Finally, an optimization model is constructed with the load balancing degree of the controller cluster, average total delay, and switch migration cost as the utility function, which is solved using an improved genetic algorithm wherein a similarity operator is incorporated to enhance the speed and accuracy of seeking the global optimal solution. Experimental results demonstrate that the mechanism effectively balances the load of the control plane and optimizes network performance.

Full Text

Preamble

Vol. 39 No. 9
Application Research of Computers (ChinaXiv Cooperative Journal)

Research on SDN Multi-Controller Load Balancing Mechanism Based on Improved Genetic Algorithm

Xu Aixin[1],[2], Sun Shimin[1],[2]†, Wang Xiaofan[1],[2], Xu Guowei[1],[2], Wang Meiyu[1],[2]
(1. School of Software, Tiangong University, Tianjin 300387, China;
2. Tianjin Key Laboratory of Autonomous Intelligent Technology & System, Tianjin 300387, China)

Abstract: To address the problem of multi-controller load imbalance in Software-Defined Networking (SDN), this paper proposes a Non-cooperative Game Theory-based Master Controller Reselection model for load reduction (GTMCR). First, dynamic thresholds are employed to identify overloaded controllers. Second, a priority-based decision mechanism for migrating switches is adopted. Finally, an optimization model is constructed with the load balancing degree of controller clusters, average total delay, and switch migration cost as utility functions, which is solved using an improved genetic algorithm. A similarity operator is introduced to enhance the speed and accuracy of seeking the global optimal solution. Experimental results demonstrate that this mechanism effectively balances the control plane load and optimizes network performance.

Keywords: software defined networking (SDN); non-cooperative game; load balancing; switch migration; genetic algorithm; operator

0 Introduction

Software-Defined Networking (SDN) [1] represents a novel Internet architecture that not only achieves separation between the control and data planes but also enables flexible and convenient network configuration and management through centralized controller control. However, due to dynamic changes in network scale and traffic volume, controllers may experience load imbalance, leading to network malfunction. Overloaded controllers may also become vulnerable to malicious traffic attacks, affecting normal network communication and posing new challenges to stable and efficient network services. Therefore, research on load balancing among multiple controllers is of significant importance.

Currently, multi-controller deployment is mostly accomplished through static approaches [2,3]. In real communication environments, however, network traffic varies dynamically. Consequently, designing dynamic controller-switch mapping mechanisms to reduce controller response time and better utilize controller resources plays a vital role in improving the scalability and reliability of the SDN control plane. Liu et al. [4] proposed a phased dynamic load balancing strategy targeting the controller load imbalance problem caused by dynamic traffic changes in SDN, which focused solely on minimizing migration cost without considering other optimization objectives. Huang et al. [5] introduced a Clustering-based Genetic Algorithm with Cooperative Clusters (CGA-CC) to address dynamic controller deployment, employing a scheduling algorithm based on request allocation probabilities across all controllers to balance the trade-off between scheduling performance and scalability. However, they did not provide detailed description of metric weight allocation, resulting in limited load balancing effectiveness. Adekoya et al. [6] developed an Improved Switch Migration Decision Algorithm (ISMDA) that selects optimal controller deployment schemes through migration models based on traffic capacity constraints for migrating

switches, though further experiments are needed to validate the algorithm' s feasibility and effectiveness.

Comprehensive consideration of current controller deployment schemes [7] reveals that load balancing is achieved through switch migration among multiple controllers. The main challenges are: from the controller' s perspective, how to determine the overloaded state and which switch to migrate to achieve master-slave controller remapping for control plane load balancing; from the switch' s perspective, which target controller to map to for load balancing and how to design dynamic mapping objectives between controllers and switches.

1 Related Work

In recent years, regarding the identification of overloaded controllers in multi-controller architectures, Lan et al. [8] proposed a dynamically adaptive load balancing strategy with predefined load weights, which significantly reduced flow overhead but exhibited poor performance in controller response time. Mokhtar et al. [9] presented a multi-threshold load balancing switch migration scheme that introduced indicators for transmitting load information and achieved continuous load balancing among controllers through gradual dynamic threshold adjustment. However, the indicators increase computational costs for load balancing and suffer from poor real-time performance.

The selection of switches for migration mostly adopts random principles, with traditional approaches failing to consider the load of the destination domain, easily causing secondary overload. Therefore, Xiao et al. [10] proposed a Decision-Making Scheme of Switch Migration (DMSSM) for the SDN control plane, which introduced a migration benefit model to address migration decision problems by forming a set of switches in the migration domain and measuring switches to be migrated based on load balancing rate and transmission delay, though total migration delay was not considered. Sahoo et al. [11] designed a load balancing algorithm that minimizes load differences among controllers to select switches for migration, but it failed to account for controller load capacity. Controllers with small capacity may become overloaded after migration, potentially triggering erroneous switch migrations. Zhong et al. [12] proposed a prediction-based dual-weight switch migration scheme for SDN load balancing that uses historical traffic load data to predict future traffic loads and reduces switch migration frequency, but traffic prediction requires additional processing time, undoubtedly increasing computational costs.

Regarding the deployment of target controllers for migrating switches, Yuan et al. [13] proposed a Genetic Algorithm-based Controller Placement Approach (GAPA) to solve controller configuration problems. This method demonstrates good performance in reducing average propagation delay between SDN switches and controllers while achieving better balance among controllers. However, its crossover and mutation genetic operators employ traditional fixed values, causing it to miss optimal deployment schemes and resulting in suboptimal optimiza-

tion performance. Mohanty et al. [14] proposed an effective genetic algorithm-based controller placement method (Proposed GA) that minimizes propagation delay and optimization cost to find optimal controller positions with minimal cost, but the algorithm suffers from slow convergence, leading to excessively long deployment times and inefficient mapping strategies. Ibrahim et al. [15] presented a Genetic Algorithm (GA) to determine the optimal positions and number of controllers, minimizing delay as the optimization objective while reducing iteration counts and ensuring load balancing among controllers, though the selection operator in the algorithm yields less reasonable strategies.

For multi-objective optimization in selecting target controllers, Schütz et al. [16] proposed a comprehensive mathematical formalization for controller deployment that determines switch-controller mapping relationships based on propagation delay and controller capacity while maintaining balanced load distribution among controllers. However, their approach only optimizes average propagation delay without considering transmission and processing delays. Ibrahim et al. [17] introduced a Greedy Random Search (GRS) algorithm to solve dynamic allocation problems between nodes and controllers, achieving maximum resource utilization for load balancing through controller position and quantity allocation. Such strategies require correct controller placement to improve network reliability and cost-effectiveness but fail to consider factors beyond resource utilization. Zhong et al. [18] proposed the Assessing Profit of Prediction (APOP) scheme, a load balancing strategy for multi-controller control planes based on overload state prediction and profit evaluation, but it only considers load balancing degree without optimizing other network performance factors such as communication delay.

Based on the current literature survey, to minimize computational cost and response delay as much as possible, this paper adopts a dynamic threshold method for overloaded controller selection. The selection of switches for migration comprehensively considers traffic variation, load capacity, and migration distance, employing a priority-based approach. For target controller selection, an improved genetic algorithm is used, which introduces a similarity operator to accelerate convergence while adjusting selection, crossover, and mutation operators to ensure optimal deployment strategies. By combining migrating switches with master controller reselection and considering the impact on network performance before and after migration, the designed optimization objectives comprehensively incorporate the following metrics: uniform load across controllers, maximized resource utilization, reduced total delay between control and data planes, and minimized switch migration cost. Through this switch migration approach for control plane load distribution, this work provides solutions and modeling insights for multi-controller load balancing strategy research.

2.2 Non-Cooperative Pure Strategy Nash Equilibrium Model

Non-cooperative game theory addresses how to make decisions to maximize one' s own benefit in situations where interests are mutually influential, i.e., the strategy selection problem. To solve the problem of determining target controller deployment strategies that maximize utility function values, this paper constructs a non-cooperative game model for SDN networks. The game model can be represented as a triple tuple, with the following three elements:

a) **Participants Q**: Slave controllers that are not overloaded serve as participants, excluding overloaded controllers. In Equation (2), Cr represents the set of overloaded controllers.

b) **Strategy Set P**: A pure strategy refers to selecting a strategy with probability 0 or 1, meaning participants use only one strategy from the strategy set. Overloaded controller Cr migrates switches Si (i (1,m)) under its control domain to target controller Ck. The game proceeds among the controller set Q. After selecting the game opponent Ct, each participant has two pure strategies, represented by Equation (3).

c) **Utility Function U**: The utility of all participants under a specific strategy combination. It includes controller load, total delay between switches and controllers, and switch migration cost.

Definition 1: Utility Function Based on Controller Load Balancing Degree. In the interaction packets communicating with controllers, data packets from various switches dominate, such as Packet_{In} messages. Therefore, this paper uses the number of Packet_{In} messages processed by controllers to calculate controller load. Rj represents the maximum number of Packet_{In} packets processed by controller Cj per unit time, ri represents the number of Packet_{In} packets generated by switch Si per unit time, and M represents the number of switches to be migrated. Controller load can be calculated using Equation (4).

The average load degree of controllers can be represented by Equation (5). To reflect the load balancing distribution of the SDN control plane, the load balancing degree u1 of the controller cluster is adopted as the metric, obtaining load difference among controllers through standard deviation, as shown in Equation (6).

2.1 SDN Distributed Network Model

The network topology is represented as G(V,E), where V and E denote the sets of nodes and links, respectively. The n controllers and m switches in the network are represented as C = {C_1, C_2, ···, C } and S = {S_1, S_2, ···, S }, respectively. The mapping relationship between switches and controllers is represented by a mapping matrix (1), which indicates the link conditions between network

nodes. According to the controller' s management scope, the network is divided into several subdomains, each including a controller and its managed switches. In a multi-controller architecture, based on different switch access permissions, the OpenFlow 1.3 protocol defines three types of controllers with different role functions: equal controllers, master controllers, and slave controllers. Among them, master and equal controllers are responsible for managing and controlling switches, while slave controllers can only read switch status without managing them.

Definition 2: Utility Function Based on Average Delay Between Switches and Controllers. The delay between switches and controllers determines the real-time performance of decision distribution and affects overall network performance. The delay of data packets between switches and controllers mainly includes propagation delay and queuing delay during network congestion. The average total delay from switches to controllers is used as the measurement standard, as shown in Equation (7), where Tij represents the total delay from switch Si to controller Cj.

That is, the utility function values all satisfy the inequality.

3 Controller Load Balancing Model

Definition 3: Utility Function Based on Average Switch Migration Cost. During switch migration, achieving control right transfer requires a complex role conversion process. When the switch to be migrated is determined, the controller installs Flow_{Mod} rules within the domain to the migrating switch, incurring overhead PI as shown in Equation (8), where τ represents the average size of flow table installation packets (including Flow_{Mod} message packets, etc.), and lij represents the shortest reachable inter-node link length from switch Si to target controller Cj.

The communication cost of switch migration arises from the cost of deploying flow policies, primarily influenced by the number and size of messages that need to be migrated, including the communication cost PC between migrating switches and incoming/outgoing controllers, as shown in Equation (9). represents the average size of switch communication message packets (including Packet_{In} packets, etc.), and $a_0$ij and aij represent the mapping relationships between switch Si to be migrated and target controller Cj before and after migration, respectively.

When the migrating switch sends flow requests to the target controller, migration request cost PM is generated, as shown in Equation (10), where represents the average size of switch migration request message packets (including Role-Request message packets, etc.).

The number of switch migrations can be represented by Equation (11). This paper comprehensively considers migration rule flow table installation cost PI,

communication cost PC, and migration request cost PM. Equation (12) represents the average migration cost u3.

If u1, u2, and u3 are taken as objective functions, the problem of finding the optimal controller-switch mapping matrix relationship can be transformed into a multi-objective optimization model. The optimization model in Equation (13) seeks the optimal solution to minimize ui values, i.e., minimizing load differences among controllers, total delay between switches and controllers, and switch migration cost.

For the load-imbalanced control plane, this chapter first identifies overloaded controllers through the ODT method, then uses the PDSS strategy to select switches under overloaded controllers for migration to target controllers deployed by the GAIMO algorithm. Finally, the Game Theory-based load-reduction Master Controller Reselection model (GTMCR) performs load reduction on overloaded controllers in the controller cluster, thereby achieving control plane load balancing. The algorithm concept is illustrated in Figure 1 [Figure 1: see original paper].

3.1 Overloaded Controller Determination Strategy ODT

For overloaded controller determination, this paper designs an Overloaded Dynamic Thresholds mechanism (ODT), where the threshold LT is dynamically adjusted based on the overall control plane load status, as defined in Equation (14).

In this model, to reduce additional overhead from frequent switch exchanges within the cluster, remapping operations are not performed unless the current controller becomes overloaded. Each controller processes as many packets as possible without exceeding its load capacity, as this increases its own throughput while reducing load increases for other controllers, conforming to the Nash equilibrium principle where each participant' s strategy is the optimal response to other participants' strategies. Therefore, this game possesses a pure strategy Nash equilibrium.

This paper uses Pi to represent a specific strategy of a participant, and P-i to represent the combination of strategies of all other participants except participant i, specifically denoted as P-i = (P_1, ⋯, P $_1$, P $_1$, ⋯, P), representing a specific strategy combination of all participants. When the game reaches stability, the Nash equilibrium state is represented by P. *For a game with n participants, under the condition that other participants' strategies remain unchanged, for any participant i, its strategy is the optimal response to other participants' strategy combinations, which can be expressed as u (P* , P *) ≤ u (P , P)*. In Equation (15), represents the threshold value for determining overload, LC represents the load condition of controller C , M = 0 indicates non-overloaded status, while M = 1 indicates overloaded status, triggering GTMCR for load balancing. The controller status flag field is shown in Equation (15).

3.2 Migration Switch Selection Strategy PDSS

To quickly reduce load on overloaded controllers, this paper proposes a Priority Decision Switching Strategy (PDSS). If a large number of new requests arrive after the master-slave controller relationship change, new flow entries will be generated, sending more Packet_{In} requests to the new destination controller and increasing service processing burden. Therefore, switches with smaller traffic changes have higher migration priority. Additionally, switches with larger loads can complete load reduction through a single migration, preventing excessive migration costs caused by large migration quantities. Furthermore, to reduce communication costs between switches to be migrated and overloaded controllers, switches closer to target controllers have higher migration priority. Consequently, priority is given to migrating switches with small traffic changes, high load proportion, and short distance to target controllers.

Equation (16) represents the control resource Xij occupied by Si on Cj, where the number of events of Si accessing Cj is denoted by λ(Si), and Bij(si) represents the total number of hops between Si and its master controller Cj. Therefore, Equation (17) defines the priority Prir of migrating Si to Cr, where dir represents the number of hops between them, and Vi represents the flow change rate of Si. Equation (18) shows the selected migration switch Si.

3.3 Controller Decision Mechanism GAIMO Based on Improved Genetic Algorithm

The Genetic Algorithm (GA) [19] is a heuristic search algorithm commonly used to solve optimization problems. It employs probabilistic optimization search to adaptively adjust and guide the optimization search space. This section proposes the Genetic Algorithm for Improved Multi-objective Optimization (GAIMO), which uses genetic algorithms for global search of network load balancing problems, then performs further local optimization in the vicinity of the global optimum to ultimately find the optimal solution for network load balancing.

3.3.1 GAIMO Algorithm

The GAIMO algorithm incorporates the concept of a similarity operator into the basic genetic algorithm and adopts a dual-mode selection operator combined with self-adjusting crossover and mutation probabilities. The algorithm process includes: individual encoding, population initialization, individual evaluation, selection, similarity judgment, crossover, mutation operations, and population iteration replacement, with specific settings as follows:

All switches and controllers in the network are represented by an m×n dimensional matrix. Random sequence initialization is used to generate individuals with certain differences, ensuring uniformity while improving population diversity. The individual fitness function distinguishes good from bad individuals

in the population, where larger fitness values indicate better controller performance. For multi-objective optimization problems, the deviation standardization method (Min-max Normalization) is adopted to linearly transform original data through conversion functions. The fitness function is shown in Equation (19), where uimax and uimin are the maximum and minimum individual values. For the multi-objective fitness function in Equation (20), weight factors w1, w2, and w3 are employed.

The selection operator chooses individuals for creating the next generation. This paper' s selection operator uses a 5% elite strategy to first select the best individuals for inheritance to the next generation, combined with the roulette wheel method to ensure parent diversity. Equation (21) represents the probability of individual xi being selected.

Crossover aims to enable offspring to inherit excellent genes from the previous generation. Based on a fixed probability Pc' , a crossover probability related to evolutionary generations is set to accelerate population convergence and speed up the search for regions where excellent populations are located. Since excessively large Pc values can quickly destroy gene strings with high environmental adaptation values, the Pc value gradually decreases with increasing genetic generations. This also solves the problem of excessively small values slowing down the search speed. Increasing crossover points reduces performance. Therefore, to maintain diversity, this paper adopts a two-point crossover operator (Equation (22)), where Gen represents the maximum iteration count and i represents the current iteration number.

To maintain population genetic diversity and avoid falling into local maximum traps, position-selective mutation and adaptive mutation probability Pm are adopted (Equation (23)). This approach' s advantage lies in increasing mutation probability for inferior individuals while decreasing it for excellent individuals. Therefore, the original mutation probability Pm' is reduced, performing small-step large-probability mutations around excellent populations. This both prevents the algorithm from falling into local minima and greatly reduces the probability of generating useless inferior solutions again, accelerating convergence speed. As the evolutionary process progresses, the probability gradually decreases and stabilizes, avoiding impact on later-stage algorithm stability that could cause prolonged convergence time and improving computational efficiency. Here, f' , fmax, and favg represent the current, maximum, and average fitness values, respectively.

Through continuous population replacement, the algorithm eventually converges to the global optimal solution. The algorithm flowchart is shown in Figure 2 [Figure 2: see original paper].

3.3.2 Implementation of GAIMO Algorithm

Since inbreeding crossover increases unnecessary iterative calculations [20], a similarity operator is introduced based on the traditional genetic algorithm.

The Hamming distance GHij measures the similarity between two individuals about to undergo crossover, referring to the number of corresponding bits that differ between two strings i and j of the same length with base a. Therefore, similarity assessment is first performed on crossover individuals, and crossover opportunities are granted only when similarity is below the threshold. This minimizes iteration counts as much as possible, ensuring solution accuracy while accelerating convergence speed.

The proposed GAIMO algorithm optimizes the mapping relationship between switches and controllers based on genetic algorithms. First, the input parameters required for algorithm execution are determined, then binary encoding design is performed for the population, each individual is initialized, and fitness function values are calculated for every individual in the initial population (lines a~b). Next, genetic algorithm operator operations begin (lines c~s), starting with selection operator operations using a combination of elite strategy and roulette wheel selection (lines e~j). This paper proposes adding a similarity operator for similarity condition judgment; if similarity is below the threshold, adaptive two-point crossover operation is performed (lines k~n). Then genetic operations with adaptive mutation probability are conducted (lines o~p). After forming a new population (line q), the above operators continuously iterate for population replacement, and the final optimal individual represents the reselected master controller for the switch Si to be migrated (line r). The algorithm description is as follows:

Algorithm 1: GAIMO Algorithm
Input: Network parameters, maximum iteration count Gen, population size N, crossover probability crossover_p, and mutation probability mutate_p
Output: Load balancing degree, average delay between switches and controllers, switch migration cost overhead; fitness function value of the optimal individual, mapping matrix of switches to controllers for the optimal individual

3.4 Master Controller Reselection Model GTMCR Based on Non-Cooperative Game Load Reduction

To verify the effectiveness and feasibility of the proposed GAIMO algorithm, during GTMCR algorithm execution, all controllers are first traversed to monitor their loads. Overloaded controllers are identified through dynamic load thresholds, and the set of controllers requiring switch migration is determined, thereby establishing the game domain for target controllers (lines a~h). Then, based on switch migration priority, the switch with the highest priority under the overloaded controller' s management is selected for migration (lines l~n). Next, the master controller for the switch to be migrated is reselected. In the non-cooperative game among controllers, the pure strategy Nash equilibrium state for target controllers is sought, and the target controller for migration is obtained through the aforementioned GAIMO algorithm. The master-slave controller roles are then switched to achieve load reduction for overloaded controllers (lines o~r). Finally, new load balancing is achieved, and the load bal-

ancing degree of the control plane after migration is obtained (line u). Based on the migration situation, a new SDN network topology is constructed. The algorithm description is as follows:

Algorithm 2: GTMCR Algorithm
Input: Network status parameters, such as number of switches m and controllers n, controller Cj processing rate Rj, switch Si request rate ri, delay T and distance D between controllers and switches, and other related parameters
Output: Overloaded controllers in the network, load balancing degree of controller cluster, switches requiring migration in the network

4.1 Experimental Environment Setup

The experimental environment is configured with an Intel i5 processor, 4GB memory, 3.40GHz CPU, Windows 10 operating system, using MATLAB R2017b as the simulation tool. Controller capacity is set to 10MB, and delay and link distance between switches and controllers are derived from experimental topology. Experimental simulation parameter values refer to traffic characteristic values [21], with controllers processing approximately 10,000 Packet_{In} flow request packets per unit time. The experiments adopt network topologies with certain scale differences from the Internet Topology Zoo [22], with specific network information shown in Table 1 .

Table 1: Node Link Information for Different Network Sizes

Topology	Nodes	Links
OS3E		
Ntelos		
Interllifiber		
Interoute		

4.2 Performance Evaluation

This section compares the GAIMO algorithm with existing algorithms for solving controller load balancing problems—GAPA [13], Proposed GA [14], and GA [15]—in terms of network load balancing degree, average total delay, and migration cost. The fixed crossover probability is set to 0.4, mutation probability to 0.04, and iteration count to 200. Fitness function values are normalized, and to emphasize important factors affecting the control and data planes, the weight values w1, w2, and w3 are selected as 0.5, 0.3, and 0.2, respectively.

4.2.1 Controller Load Balancing Degree Comparison

At the start of the experiment, each switch sends data packets to the 5 controllers deployed in the OS3E network, dividing the entire network into five subdomains. After 10 seconds, all controllers obtain initial state values. At

this point, controller loads are relatively balanced with a load ratio of 2:4:3:2:3 (Figure 3 [Figure 3: see original paper]). Subsequently, the interaction message volume between switches and controllers is increased to gradually raise controller loads. The figure shows that controllers C2 and C4 exceed the threshold line and become overloaded successively. At the 40-second mark, the load ratio reaches 10:13:11:14:8, and the load balancing degree calculated using Equation (6) is 2.1354. Consequently, the GTMCR mechanism is triggered to perform switch migration. Afterward, load differences among controllers converge significantly. The switch migration results show that some switches under C2 are migrated to C1, and some under C4 are migrated to C5, resulting in a post-migration load ratio of 23:23:22:22:22 and a load balancing degree of 0.4899, achieving a new balance in the control plane load.

Figure 4 [Figure 4: see original paper] compares the performance of GA, GAPA, Proposed GA, and GAIMO algorithms in controller load balancing fitness. The vertical axis represents the controller load balancing fitness function value, reflecting the load balancing condition of the controller cluster. Higher fitness values indicate better controller load balancing performance. To verify the load balancing effectiveness of the four algorithms on the controller cluster, the OS3E network topology is adopted. All four algorithms show improved fitness function values, indicating that they can achieve load balancing to some extent through switch migration under load imbalance conditions. GA and GAPA exhibit large fluctuations during the 200-generation evolutionary process with poor convergence due to fixed crossover and mutation probabilities. Proposed GA' s load balancing degree fitness value improves from 0.813 to 0.915. In contrast, GAIMO converges the fastest, with its load balancing degree fitness value improving from 0.813 to 0.95, demonstrating that GAIMO more effectively balances control plane loads.

4.2.2 Average Total Delay Between Switches and Controllers

To compare the optimization effectiveness of total communication delay between switches and controllers under different algorithms, the OS3E network is used to test the optimization effect of delay fitness functions. Figure 6 [Figure 6: see original paper] shows that the highest fitness function values of Proposed GA, GAPA, and GA algorithms can only converge to 0.85, all lower than GAIMO, indicating that the mapping schemes found are not optimal. GAIMO not only converges fastest but also finds the optimal solution, better reducing total migration delay. In terms of average delay control, GAIMO outperforms other algorithms with more stable optimization effects, significantly reducing average delay between switches and controllers, more effectively decreasing communication overhead, and ensuring network communication quality.

4.2.3 Switch Migration Cost

During switch migration, numerous communication message packets are generated. To achieve controller plane load balancing while minimizing migration

cost, the OS3E network is used to test migration costs under different algorithms. The test results are shown in Figure 7 [Figure 7: see original paper]. GA and Proposed GA algorithms miss the optimal solution due to premature convergence in their selection strategies, while GAPA algorithm never converges because its fixed crossover and mutation operators cannot adjust according to iteration status. GAIMO converges faster and more easily finds the optimal solution, with the final objective function value reaching 0.9. This is because the improved operators dynamically adjust based on iteration count and fitness function value changes, achieving the goal of more efficiently finding the global optimal solution. GAIMO' s migration strategy demonstrates the best performance, significantly reducing the cost of migrating multiple switches in the network and proving the effectiveness of using switch migration methods to improve network performance.

4.2.4 Multi-Objective Performance Optimization

The multi-objective optimization fitness function value reflects controller load balancing conditions, where higher values indicate better network performance for controller load balancing. As shown in Figure 8 [Figure 8: see original paper], starting from generation 20, GAIMO' s optimal solution fitness value is superior to other algorithms with the fastest convergence speed. This is because GAIMO employs a similarity operator to avoid inbreeding in the next generation, reducing repeated and invalid crossovers and decreasing time complexity. GAIMO stabilizes and finds optimal deployment from generation 50 onward. This is due to the adaptive crossover probability considering iteration count and the mutation probability considering fitness function value preferences, with mutation occurrence decreasing as genetic generations and fitness function values increase. GAIMO's final convergence value is the highest, indicating that its elite strategy prevents loss of optimal solutions and ensures population evolution toward optimality, stabilizing at 0.94. Therefore, GAIMO better balances surging traffic loads while improving control plane robustness.

5 Conclusion

This paper constructs a multi-objective optimization model GTMCR, employing the dynamic threshold ODT algorithm for overloaded controller identification, the priority-based PDSS algorithm for migration switch decision-making, and setting corresponding constraints. The improved genetic algorithm GAIMO is used to obtain the optimal switch migration strategy. Experimental results demonstrate that GAIMO more effectively improves convergence speed, finds optimal deployment fastest, reasonably balances controller cluster load balancing degree, and enhances global network performance. Future work will consider fault tolerance issues such as packet loss during controller-switch and inter-switch communication in the remapping process. When controllers are severely overloaded or fail, besides relying on switch migration, additional controllers may need to be added to further study load balancing strategies under

different conditions.

References

[1] Isong B, Molose R R S, Abu-mahfouz A M, et al. Comprehensive Review of SDN Controller Placement Strategies [J]. IEEE Access, 2020, 8: 2602-2617.

[2] Hamdan M, Hassan E, Abdelaziz A, et al. A comprehensive survey of load balancing techniques in software-defined network [J]. Journal of Network and Computer Applications, 2021, 174: 102856.

[3] Alowa A, Fevens T, Khamayseh Y. Survival backup strategy for controller placement problem in Software Defined Networking [J]. Computer Communications, 2022, 185: 104-117.

[4] Liu Yi, Li Kaixin, Li Guoyan, et al. Dynamic load balancing strategy based on SDN [J]. Application Research of Computers, 2020, 37 (10): 3147-3152.

[5] Huang V, Chen G, Zhang P, et al. A Scalable Approach to SDN Control Plane Management: High Utilization Comes With Low Latency [J]. IEEE Trans on Network and Service Management, 2020, 17 (2): 682-695.

[6] Adekoya O, Aneiba A, Patwary M. An Improved Switch Migration Decision Algorithm for SDN Load Balancing [J]. IEEE Open Journal of the Communications Society, 2020, 1: 1602–1613.

[7] Shirmarz A, Ghaffari A. Taxonomy of controller placement problem (CPP) optimization in Software Defined Network (SDN): a survey [J]. Journal of Ambient Intelligence and Humanized Computing, 2021, 12 (12): 10473–10498.

[8] Lan W, Li F, Liu X, et al. A Dynamic Load Balancing Mechanism for Distributed Controllers in Software-Defined Networking [C]// the 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA) . Changsha: IEEE, 2018: 259–262.

[9] Mokhtar H, Di X, Zhou Y, et al. Multiple-level threshold load balancing in distributed SDN controllers [J]. Computer Networks, 2021, 198: 108278.

[10] Xiao H, Hu B, Zhou L, et al. DMSSM: A Decision-Making Scheme of Switch Migration for SDN Control Plane [C]// IEEE the 7th International Conference on Computer Science and Network Technology (ICCSNT) . Dalian, China: IEEE, 2019: 348-352.

[11] Sahoo K S, Puthal D, Tiwary M, et al. ESMLB: Efficient Switch Migration-Based Load Balancing for Multicontroller SDN in IoT [J]. IEEE Internet of Things Journal, 2020, 7 (7): 5852-5860.

[12] Zhong H, Xu J, Cui J, et al. Prediction-based dual-weight switch migration scheme for SDN load balancing [J]. Computer Networks, 2022, 205: 108749.

[13] Yuan T, Huang X, Ma M, et al. Balance-Based SDN Controller Placement and Assignment with Minimum Weight Matching [C]// IEEE International Conference on Communications (ICC) . Kansas City, MO: IEEE, 2018: 1-6.

[14] Mohanty S, Priyadarshini P, Sahoo S, et al. Metaheuristic Techniques for Controller Placement in Software-Defined Networks [C]// TENCON- IEEE Region 10 Conference (TENCON) . Kochi, India: IEEE, 2019: 166-170.

[15] Ibrahim A A Z, Hashim F, Sali A, et al. A Modified Genetic Algorithm for Controller Placement Problem in SDN Distributed Network [C]// the 26th IEEE Asia-Pacific Conference on Communications (APCC) . 2021: 178-183.

[16] Schütz G, Martins J A. A comprehensive approach for optimizing controller placement in Software-Defined Networks [J]. Computer Communications, 2020, 159: 198-205.

[17] Ibrahim A A Z, Hashim F, Noordin N K, et al. Heuristic Resource Allocation Algorithm for Controller Placement in Multi-Control 5G Based on SDN/NFV Architecture [J]. IEEE Access, 2021, 9: 2602-2617.

[18] Zhong H, Fan J, Cui J, et al. Assessing Profit of Prediction for SDN controllers load balancing [J]. Computer Networks, 2021, 191: 107991.

[19] Katoch S, Chauhan S S, Kumar V. A review on genetic algorithm: past, present, and future [J]. Multimedia Tools and Applications, 2021, 80 (5): 8091-8126.

[20] Bao L, Zhang L, Sun T. Research on assembly line scheduling based on small population adaptive genetic algorithm [C]// the 6th International Conference on Intelligent Computing and Signal Processing (ICSP) . 2021: 166-170.

[21] Isyaku B, Mohd Zahid M S, Bte Kamat M, et al. Software Defined Networking Flow Table Management of OpenFlow Switches Performance and Security Challenges: A Survey [J]. Future Internet, 2020, 12 (9): 147.

[22] Knight S, Nguyen H X, Falkner N, et al. The Internet Topology Zoo [J]. IEEE Journal on Selected Areas in Communications, 2011, 29 (9): 1765-1775.

Author Contact Information:

[23] Xu Aixin
Tiangong University, No. 399 Binsui West Road, Xiqing District, Tianjin 300387
13230847811, 13231568733@163.com

[24] Sun Shimin (Corresponding Author)
Tiangong University, No. 399 Binsui West Road, Xiqing District, Tianjin 300387
18722128913, sunshimin@tiangong.edu.cn

[25] Wang Xiaofan
Tiangong University, No. 399 Binsui West Road, Xiqing District, Tianjin 300387
15122708693, wxf241382009@163.com

ChinaRxiv 中X

[26] Xu Guowei
Tiangong University, No. 399 Binsui West Road, Xiqing District, Tianjin 300387
17519301800, 1921994090@qq.com

[27] Wang Meiyu
Tiangong University, No. 399 Binsui West Road, Xiqing District, Tianjin 300387
17835204887, 1714116205@qq.com

Note: Figure translations are in progress. See original paper for figures.

Source: ChinaXiv —Machine translation. Verify with original.