International Journal Of
Industrial Electronics Control and Optimization
Print ISSN: 3517-2645
Online ISSN: 3568-2645
Homepage: https://ieco.usb.ac.ir/

Improving Delay In SDNS By Metaheuristic Controller Placement

Maedeh Abedini Bagha1 | Kambiz Majidzadeh2 | Mohammad Masdari3| Yousef Farhang4

Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran.[1,2,3]
Department of Computer Engineering, Khoy Branch, Islamic Azad University, Khoy, Iran.[4]
Corresponding author's email: kambiz.majidzadeh@iau.ac.ir

Article Info

Article type:
Research Article

Article history:
Received: 04 Aug 2022
Received in revised form: 09 Oct 2022
Accepted: 31 Oct 2022
Published online: 21Nov 2022

Keywords:
Controller placement,
Seagull optimization algorithm,
Software-defined network.

ABSTRACT

Software-defined networking is a new network model proposed to solve the complexity of traditional network problems and facilitate dynamic network operation and management. The separation of the control plane from the data plane is the main idea of software-defined networks. Controllers are the operating system of software-defined networks and are responsible for managing the entire network. It is essential to locate controllers appropriately to have a balanced topology while guaranteeing low latency. In this work, a metaheuristic algorithm is used for controller placement. First, the problem is formulated and the network is partitioned by a clustering algorithm. Then, the seagull optimization algorithm is used to determine a suitable place for the controller in each network partition dynamically. Simulations are performed on the standard network topology from the internet topology zoo dataset to evaluate the proposed method. Simulation results reveal that the proposed method performs well in case of delay and load balancing compared with the state-of-the-art optimization algorithms.

I. Introduction

Software-defined networks (SDNs) [1] are a promising new network model proposed to solve the complexity of traditional networks [2] and facilitate the operation and management of dynamic networks [3]. The main idea in SDNs is the separation of the control plane from the data plane [4]. The control function of network components, such as switches in SDNs, is transferred to the control plane and controllers. SDNs are programable networks with many advantages [5], such as simplifying network management, improving network utilization efficiency, and supporting network innovation [6].

Network switches, also known as SDN nodes, are responsible for forwarding data based on the flow table [7], while controllers route network packets through the switches. When an unknown flow arrives at a switch, the switch sends a flow setup request to the controller, and the controller responds to the request by entering the flow in the switch's flow table [8]. Often a large-scale network with a single controller has high latency to perform control actions. In addition, the main obstacle in single controller networks is "single point failure". Also, it handles the entire network traffic, which may not be efficient. Since a single controller is responsible for all control activities in the entire SDN, any network failure will significantly impact network performance [9]. Several controllers have been distributed in the network to overcome these problems. The optimal placement of controllers in networks is a known problem, which is called controller placement problem (CPP). Because controllers play a vital role in increasing network performance, they must be located in optimal locations [10].

An example of the CPP in an SDN is shown in Fig. 1. The control plane consists of two controllers in this sample that can manage the entire network. In an SDN with multiple controllers, the number of controllers is usually determined by network scale and traffic.

The CPP unveils several challenges that must be considered. The optimal place of controllers is one of the CPP challenges as it widely impacts several parameters like delay, reliability,

Copyright © 2022 Abedini et al. Published by University of Sistan and Baluchestan. This work is licensed under a Creative Commons Attribution-Noncommercial 4.0 International license. (https://creativecommons.org/licenses/by-nc/4.0/). Non-commercial uses of the work are permitted, provided the original work is properly cited
DOI: 10.22111/IECO.2022.42638.1436

and load balancing [11]. The solutions should consider the controllers' capacities, the load distribution between the controllers, and the delay. Just as delay is the most crucial challenge in all networks [12], it is also crucial in SDN, and the propagation delay between the controllers and the switches must be considered in CPP.

The CPP is difficult due to its NP-hard nature [13]. The conventional formulation for CPP optimizes the objective function by linear programming (LP). But, LP does not run effectively on large networks with many nodes and links. Since real SDNs are often dynamic with variations (e.g., the number of users, topology, device failure, etc.), computation time and resources are constraining factors [11]. Therefore, it is suggested to use metaheuristic techniques to solve it.

Seagull Optimization Algorithm (SOA) [14] is a new population-based bio-inspired algorithm. Some of the advantages of metaheuristic algorithms including SOA are flexibility, deviation-free, simplicity, understandability, and avoidance of local optima [15]. Metaheuristic algorithms focus on exploring the workspace to reduce the stickiness of a local solution. Local solutions are not the right solution to any problem [16]. SOA is flexible enough to solve different problems of different natures. It uses the randomness of variables to solve problems and eliminate computation to extract search space, making it more applicable to existing problems [14].

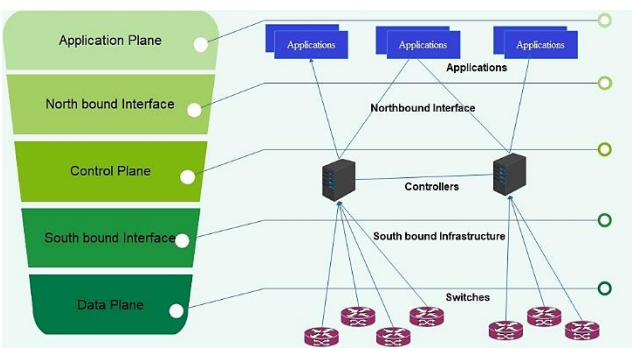


Fig. 1. An example of the CPP in SDN

In this work, we find the appropriate location of the controllers in the network. This research formulates CPP based on the load balancing in controllers, controller-to-switch latency, and inter-controller latencies.

Since CPP is NP-hard and to solve such problems, an optimal solution cannot be obtained in a reasonable time [15], researchers have developed optimization algorithms based on metaheuristics. Metaheuristic techniques are suitable for solving complex problems and are widely used [17]. So, we use SOA to determine the location of the controllers. To the best of the authors' knowledge, this is the first time that SOA is used to solve the CPP. Finally, the proposed method is evaluated using the Internet topology zoo dataset. The main contributions of this research can be summarized as follows:

1. Formulating the CPP as an optimization problem aimed to minimize the delay and improve load balancing
2. Partitioning the network by Fuzzy C-Means (FCM) clustering algorithm
3. Determining appropriate places for controllers by SOA

The rest of the paper is organized as follows. Section 2 presents a literature review of the CPP. Section 3 defines and formulates the CPP. The proposed method, clustering network by FCM, and controller placement by SOA are presented in Section 4. The proposed method is evaluated in Section 5. Finally, Section 6 concludes the paper.

II. Related Works

Separating the data and control planes in the SDN architecture forces operators to design the control plane, so the design of the control plane is a hot topic in SDN. CPP has been studied from various perspectives in previous research, such as network performance [18], [19], [20], [21], deployment scenarios [22], [23], [24] etc., focusing on optimizing network performance. The following is an overview of research associated with CPP.

CPP was first introduced by [18] in 2012. The authors studied CPP to minimize the propagation latency from the switches to the dedicated controller. Their approach included a large class of localization problems corresponding to a K-center problem to minimize the average propagation delay.

Controller load balancing is another critical performance metric on CPP. Dixit et al. [22] proposed a flexible distribution of controllers to maintain the number of controllers by changing the controllers' number according to the SDN traffic. However, these methods improved only one parameter, like propagation delay or load balancing.

Authors in [25] studied a multi-objective genetic algorithm to produce an approximate Pareto optimal frontier on large-scale networks. They considered load imbalance, latency, and reliability of network parameters. Also, they introduced inverted generational distance (IGD) as a performance metric to measure the distance between the estimated Pareto frontier and the optimal Pareto frontier.

In [26], the authors focus on CPP in software-defined vehicular networks (SDVN) and consider the dynamicity of vehicular networks' topology. They proposed a dynamic placement method based on integer linear programming (ILP) that adaptively adjusts the number and location of controllers according to road traffic fluctuations. But, ILP does not run effectively on large networks with many nodes and links.

In [27] and [28], the authors try to solve CPP by partitioning the network. In [27], a hierarchical K-means algorithm is proposed, and in [28], a network partitioning technique with an optimized K-means algorithm is proposed to minimize the delay. Authors in [9] introduce deep Q-network-based dynamic clustering and placement (DDCP) for dynamic clustering of SDN switches and controllers and show that it

can significantly improve response time and network resource utilization. However, deep learning requires a large amount of data and computational resources.

In [20] and [21], the authors propose a mathematical model to improve the delay in the distributed controller plane in SDN. They use two data-ownership models, single-data ownership (SDO) and multiple-data ownership (MDO), which provide consistency and eventual consistency. They show that both the switch-to-controller and the inter-controller latencies affect the response time of controllers in the SDO model. Also, the switch-to-controller latency is the only significant parameter affecting the controllers' response time in the MDO model. They propose two mathematical models to solve CPP that minimize the response time. For this purpose, they comprehensively examine all possible controller placement locations to determine Pareto optimal locations in switch-to-controller and inter-controller latencies. Still, they do not consider the capacity of controllers and load balancing of controllers.

Gao et al. [29] used particle swarm optimization (PSO) to solve CPP while considering the capacity of controllers. Gao et al. [30] presented a variant of the PSO heuristic for CPP while accounting for the load balance of controllers and latency. In [31], a discrete cuckoo search algorithm is applied to CPP. Simulation results show that the discrete cuckoo search algorithm performs well regarding latency and process time. However, it is designed for static environments and does not consider the load balancing of controllers.

In [32], a dynamic optimization algorithm is developed based on the salp swarm optimization algorithm (SSA). It dynamically checks the optimum number of controllers and the optimal assignment of switches to controllers in large-scale SDNs. It aims to improve execution time and reliability but it does not consider controllers' capacity and load balancing.

Schütz and Martins [33] proposed a formal mathematical model to reduce the controllers' propagation delay and load balance. The authors determined the location of the controllers, taking into account the capacity of the controller and the minimum number of controllers, and assigned nodes to the controllers.

Authors in [10] use Pareto integrated tabu search (PITS) algorithm for CPP. They use a graph theory approach to determine the proper number of controllers. The PITS algorithm is proposed to identify the optimal location of controllers and switch assignments. Also, a heuristic approach is proposed to perform switch migration and improve SDN's reliability.

Authors in [34] propose a new controller placement algorithm using Manta-Ray foraging optimization (MRFO) and salp swarm algorithm (SSA), but it only improves the delay parameter. They do not consider inter-controller latency either.

Authors in [35] deal with traffic and energy consumption in SDN. They design an energy-aware routing algorithm formulated as a multi-objective NP-hard for the CPP. A heuristic approach is developed to find optimal routing for traffic between controllers and the controller to switches to improve energy consumption. It can rearrange traffic to meet provisioning criteria by utilizing the capacity-aware design, but the network delay is not considered.

Authors in [24] address CPP and design a subgraph of a tee that can satisfy delay and availability constraints. Also, they consider geodiversity constraints to improve robustness to disaster-based failures. In [36], a DPC optimization algorithm is designed to solve the problem of poor aggregation effect in low-density regions. Also, the SC metric is addressed. The authors improve the BeeDPC algorithm to determine the controllers' position, simplify controller design and reduce average latency in SDN. Simulation results show that the optimized DPC algorithm can reduce the average delay. However, they do not consider the controller capacity.

Authors in [37] propose a hybrid metaheuristic algorithm for CPP. At first, the authors improve the SDN network based on graph theory. Then, the optimal controllers are selected using the firefly optimization algorithm (FA). Finally, multiple controllers are placed using a hybrid metaheuristic algorithm, which includes a harmonic search algorithm and a particle swarm optimization algorithm (HSA-PSO). Among the advantages of this method can be mentioned the improvement of latency, reliability and throughput.

Most existing research on controller placement optimizes the delay by improving switch-to-controller propagation latency, but they usually don't consider the capacity and load of the controllers. This research investigates the problem of delay minimization considering controller-to-controller and switch-to-controller latencies, controllers' capacity, and their load balance.

III. Problem Definition

In this section, the CPP treatment is formulated by the following assumptions.

(a) Each switch is assigned to precisely one controller.
(b) Each controller can respond to requests according to its capacity. Also, the sum of the requests processed by each controller at any moment must be within the processing capacity of the controller.
(c) The switch requests cannot be split to be processed by multiple controllers.
(d) The network load is dynamic and changes over time.
(e) Each controller can only be placed in one of the nodes' coordinates.
(f) No two or more controllers can share the same nodes' coordinates.

We modeled an SDN topology by a graph $G = (V, E)$ where V is the finite set of our switches or nodes and E is the set of links between them. Then, K is the number of controllers that must be placed in SDN and $K \leq |V|$.

A. End-to-End Delay

Eq. (1) shows the SDN total delay from source to destination [38] where $delay_{end\text{-}end}$ is an end-to-end delay, d_{proc} is the processing delay at each router or controller, and d_{prop} is the propagation on each link. Also, d_{trans} is the total transmission rate of each switch to the controller that shows ds2c and the delay of transmission rate from each controller to another controller, shown by d_{c2c} in Eq. (2).

$$delay_{end\text{-}end} = d_{proc} + d_{trans} + d_{prop} \quad (1)$$

$$d_{trans} = d_{s2c} + d_{c2c} \quad (2)$$

The number of network nodes is denoted by N, and $d_{v,c}$ represents the shortest path from controller $c \in C$ to node $v \in V$. Eq. (3) shows the average propagation delay for a set of controllers C [13].

$$d_{avg}(C) = \frac{1}{N} \sum_{v \in V} \min_{c \in C} d_{v,c} \quad (3)$$

and the worst-case propagation latency for a set of controllers' C[13]:

$$d_{wc}(C) = \max_{v \in V} \min_{c \in C} d_{v,c} \quad (4)$$

B. Inter-Controller Propagation Latency

The average controller-to-controller latencies characterize the average latency between any pair of controllers in the network. The average of inter-controller latencies is calculated by Eq. (5) where $d(c1, c2)$ is the delay of c1 to c2:

$$d_{c2c} = \frac{1}{K.(K-1)} \sum_{c_i, c_j \in C} d(c1, c2) \quad (5)$$

C. Switch to controller propagation latency

We need to calculate the latency of switch-to-controller propagation, too. Eq. 6 shows the average latency of switch-to-controller in SDNs where $A_{s\,c}$ is the assignment of switch s to controller c.

$$d_{s2c} = \frac{1}{|V|} \sum_{s \in S} \sum_{c \in C} d(s, c) \; A_{s,c} \quad (6)$$

D. The Load of Controller

The traffic between SDN switches is unpredictable, and there are no known traffic patterns, but traffic load can be monitored during the runtime. The processing load of controllers often consists of three main parts: a load of rule installation in each controlled switch by c, the computational load of routing new arriving flows from the outside of SDN, and the computational load of routing new arriving flows from other SDN domains. B ($f_{o,\,Si}$) shows the computational load of routing newly arrived flows at switch (S_i) from the outside of SDN with a rate of $f_{o,\,Si}$.

B ($f_{Sj,Si}$) shows the computational load due to routing at the controller of Si for flows arriving at S_i from switch S_j controlled by another SDN controller with a rate of $f_{Sj,\,Si}$.

Calculating SDN controllers' load is critical to performing path computation and depends on the arrival rate of flows through function B. The objective of this study is not the definition of function B [39].

LR in Eq. (7) shows the computational load to rules installation for switch Si at the controller [36].

$$LR = \sum_{Sj \in S} F(f_{Si,sj})/bw_{ij} + F(f_{Si,o}) \quad (7)$$

It expresses the number of flows traversing Si going out of SDN or other switches. Function F maps the flow arrival rate at Si to the computational load at the SDN controller needed for rules installation. bw_{ij} is the bandwidth of link node i to node j. Pm denotes the set of assigned switches. The overall computational load at SDN controller (LT_c) is computed by Eq. (8) [36].

$$LT_c = \sum_{Si \in Pm\,,Sj \in S} B(f_{0,Si}) + \sum_{Sj \notin Pm\,,Si \in Pm} B(f_{Sj,Si})/bw_{ji} + \sum_{Si \in Pm\,,Sj \in Pm} F(f_{Si,Sj})/bw_{ij} + \sum_{Si \in Pm} F(f_{Si,o}) \quad (8)$$

Eq. (9) shows the average load of all controllers in the entire network.

$$LT_{Avg} = \frac{1}{K} \sum_{c \in C} LT_c \quad (9)$$

IV. Proposed Method

The optimal locations of controllers change dynamically as the network load changes. Consequently, we must solve a dynamic optimization problem to find the optimal location of controllers in the network [25]. There are several switches in SDN that controllers must control. But, the problem is how switches are assigned to controllers that can provide the communication locality to decrease delay and increase load balancing at the controller level. For example, if switch No. 3 usually sends data to switch No. 8 to provide the locality of the communication, it is better that switches No. 3 and No. 8 are supported by the same controller. This research aims to find the best arrangement of switches with controllers to localize, reduce inter-controller and switch-to-controller latency, and balance the controllers' load.

The proposed method is based on SOA, which uses the seagull optimizer to find the proper location for the controllers. Firstly, the network is clustered by FCM, and then the optimal controller locations are determined in each cluster by SOA.

In the first step, the network is clustered by FCM [40]. FCM is a clustering method that allows each data point to belong to multiple clusters with varying degrees of membership. Previous research like [27] and [28] used K-means clustering to partition the network, but we use FCM because of its flexibility. This means that if a node is in one cluster in the clustering step, its cluster can be changed in the placement step, which is possible in FCM. To optimize the controller placement, SOA [14] is adopted in the next step.

After FCM clusters the network, the controllers' location in each cluster is determined using SOA. In defining the fitness function of SOA according to the research objectives, switch-to-controller latency, controller-to-controller latency, and controllers' load are considered.

A. Seagull Optimization Algorithm

We use the seagull optimizer to find the optimal place for controllers in each cluster [41]. SOA is a metaheuristic population-based algorithm that simulates seagulls' migration and attacking behaviors. Population-based metaheuristic algorithms share various advantages, including scalability, simplicity, and computational time reduction [32] .Also, population-based algorithms can find the global optimum [14]. SOA is superior to optimizers such as spotted hyena optimizer (SHO) [41], grey wolf optimizer (GWO) [42], particle swarm optimization (PSO) [43], moth-flame optimization (MFO) [44], and genetic algorithm (GA) [45] with better exploration and exploitation, less time complexity and faster execution time.

Seagulls are seabirds that are found all over the planet. Seagulls generally live in groups. They use their intelligence to find and attack their prey. They attract fish by baiting bread crumbs or earthworms. They make rain-like sounds with their feet to draw earthworms hidden beneath to the ground. The most important thing about seagulls is their migratory and aggressive behavior. Migration is defined as the seasonal movement of seabirds from one place to another to find a place with a better climate and more food resources [46].

Seagulls' behavior is illustrated as follows:

• Seagulls migrate in groups. During migration, they maintain their position to avoid colliding with each other.

• Seagulls can move towards the best seagull for survival in the group.

• Seagulls can update their initial positions based on the fittest of the seagull.

These birds often attack when they see migratory birds. They attack with a spiral shape. These behaviors can be formulated so that they can be optimized with the objective function. SOA focuses on two natural behaviors of seagulls: migration and attack.

1) Migration (exploration)

During migration, the algorithm simulates how a group of seagulls moves from one position to another. In this phase, a seagull should satisfy three conditions: 1. collision avoidance, 2. the movement direction, and 3. remaining close to the best seagull.

a) Collision avoidance:

With the help of variable *A*, the new Search Agent (SA) position is calculated, and collisions with neighbors are avoided in migration.

$$\overrightarrow{C_s} = A \times \vec{P}_s(x) \quad (10)$$

where $\overrightarrow{C_s}$ represents the position of the SA that does not collide with another SA, x indicates the current iteration, $\vec{P}s$ represents the current position of the SA, and *A* represents the movement behavior of the SA in a given search space that is calculated by Eq. (11) in the standard version of SOA.

$$A = f_c - (x \times \frac{f_c}{Max_{iteration}}) \quad (11)$$

where x=0, 1, 2, …., Max $_{iteration}$ and *A* is a variable that decreases linearly from f_c to 0, and f_c sets the frequency of using variable *A*.

b) Movement Direction:

The SAs move to the best neighbor while avoiding collisions between neighbors.

$$\overrightarrow{M_s} = B \times (\vec{P}_{bs}(x) - \vec{P}_s(x)) \quad (12)$$

where $\vec{Ms}$ represents the positions of SA $\vec{Ps}$ towards the best-fit SA $\vec{Pbs}$ (i.e., fittest seagull), the variable *B* is deployed to make balancing between exploration and exploitation randomized. Eq. (13) states *B* calculation.

$$B=2\times A^2 \times rd \quad (13)$$

where *rd* is a random number in the range of [0, 1].

c) Remaining Close to the Best Seagull:

Finally, the SA can update its position concerning the best SA.

$$\overrightarrow{D_s} = |\vec{C_s} + \vec{M_s}| \quad (14)$$

The distance between the SA and the best agent is illustrated by $\vec{D}$s.

2) Attacking (Exploitation)

When seagulls attack prey, they perform a spiral movement. This behavior in the x, y, and z planes are defined. The new position of the SA is calculated using Eq. 15 to 18. In these equations, *r* is the radius of each spiral turn, *k* is a random number in the range of 0 to 2π, *u* and *v* are constants to define the spiral shape, and *e* is the base of the natural logarithm.

$$x' = r \times \cos(k) \quad (15)$$

$$y' = r \times \sin(k) \quad (16)$$

$$z' = r \times k \quad (17)$$

$$r = u \times e^{kv} \quad (18)$$

The best solution is stored in $\overrightarrow{P_s}(x)$, and the position of other SAs is updated based on it. In SOA, like most metaheuristic population-based algorithms, a random initial population is first created. SAs update their position according to the best SA in each iteration. *A* decreases linearly from *fc* to *0*. Variable *B* is responsible for the smooth transition between exploration and exploitation

[14].

$$\vec{P_s}(x) = \left(\vec{D_s} \times \mathrm{x}' \times \mathrm{y}' \times \mathrm{z}'\right) + \vec{P}_{bs}(x) \quad (19)$$

The initial population of seagulls is vectors, along with the number of switches. For example, if the initial population is 20 seagulls and the number of switches is ten, the initial size of the matrix will be 20×10. Suppose that the algorithm's input and the number of controllers for placement are "3". In that case, the initial population should be adjusted so that only three elements to locate the controller in each vector are selected. Element number may be "1" or "0". "1" shows that this place is selected as controller placement, and other places equal "0". So, in the initial population vector, three elements will be equal to "1", and seven elements will be equal to "0". Fig. 2 shows an example of three seagulls.

The basic idea of the SOA in this study is to satisfy the controller plane's load balance, minimize the inter-controller, and switch to controller distances and latency. So, the fitness function is modeled as follows:

$$\text{Min fit}(d_{trans}, LT_{Avg}) \quad (20)$$

where *fit* is a non-linear function of deployed controllers and d_{trans} is the total transmission rate out of each switch to the controller and the delay of transmission rate out of each controller to another controller calculated by Eq. (2). LT_{Avg} is the average load of controllers calculated by Eq. (9). The fitness function aims to minimize these parameters.

Seagull 1	0	0	0	1	0	0	1	0	1	0
Seagull 2	0	0	1	1	1	0	0	0	0	0
Seagull 3	0	0	0	0	0	0	1	0	1	1

Fig.2. An example of three seagulls

V. Performance Evaluation

A simulation environment is performed for the proposed method's performance testing. We compare our proposed method (SOA) with the state-of-the-art metaheuristic algorithms like Cuckoo Search Algorithm (CSA) [31], SSA [32] , and PSO [29] on the Internet topology zoo (ITZ) dataset [47].

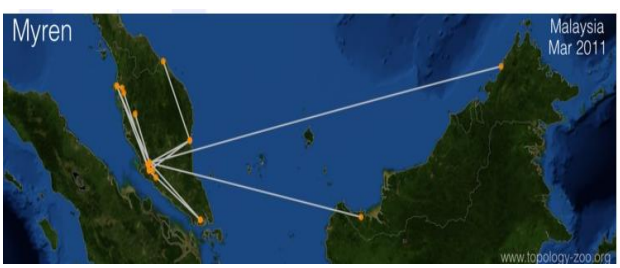




Fig. 3 Myren and Cogent network topologies [36]

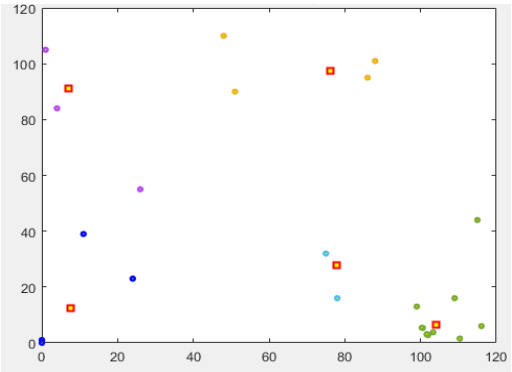


Fig. 4. The controller placement in the Myren topology

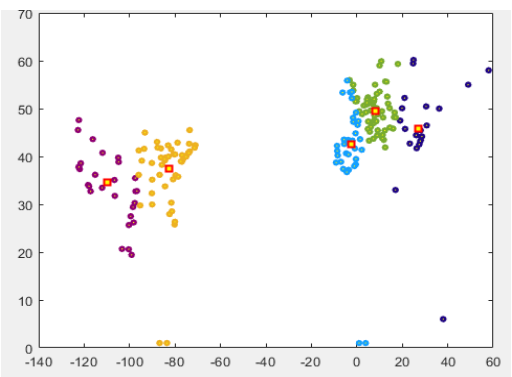


Fig. 5. The controller placement in the Cogent topology

A.Simulation Setup

The proposed method is evaluated using MATLAB 2018b over a machine equipped with an Intel Core i3 processor and 8 GB RAM. As shown in Fig. 3, the Myren and Cogent topologies from the ITZ dataset [47] are selected as small and large network topologies. The detail of these topologies is shown in Table I. Table II shows the simulation parameters. The initial number of populations in algorithms is set to 200, the iteration count is 50, and the algorithms run for ten rounds. Figs. 4 and 5 depict five controllers' placement and node assignment obtained by the proposed method. In these figures, color stars show the switches, and red squares show the controller locations.

TABLE 1.
DETAILS OF THE SOFTWARE-DEFINED NETWORKS

	Network	Type	Geo extent	Geolocation	Layer	No. switch	No. links
1	Myren	REN	Country	Malaysia	IP	36	39
2	Cogent	COM	Country+	USA, Europe	IP	197	212

TABLE 2.
THE SIMULATION PARAMETERS

Parameter/ constants	value
Initial population	200
Run time	10 rounds
Traffic	Random (0,250)
Iteration	50
PSO	$c_1 = 2.0$, $c_2 = 2.0$, w(0.9→0.4)

A. End-to-End Delay

In this section, the end-to-end delay is evaluated based on the number of packages sent. As can be seen in Figs. 6 and 7, the proposed method differs significantly from the other two algorithms in terms of the end-to-end delay. The proposed method, in most cases, has less latency than the other three methods and has made a significant improvement in reducing latency. This is because of considering the controller-to-controller and switch-to-controller latencies in the fitness function and the flexibility of FCM to handle the nodes to select the appropriate cluster.

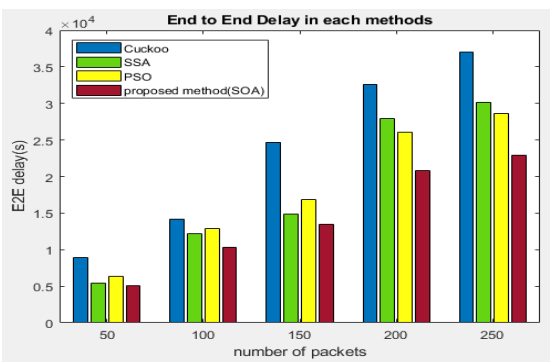


Fig. 6. End-to-end delay in Myren

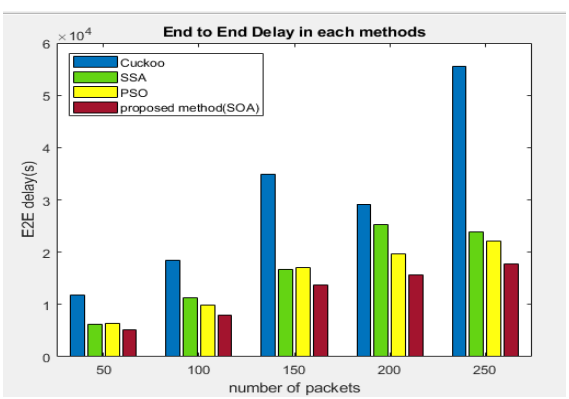


Fig. 7. End-to-end delay in Cogent

B. Load Balancing

The controllers' load is evaluated to evaluate the network load balance. For this purpose, the maximum load of the controllers and the total load of the controllers over time are investigated. Figs. (8) and (9) show the maximum load of controllers in Myren and Cogent topologies in 10 rounds, respectively. Figs. (10) and (11) display the summation of controllers load in Myren and Cogent topologies in 10 rounds. Cuckoo and SSA have the highest loads among evaluated methods, but the proposed method has good performance in this parameter, and it has a better load balance and proper load distribution on the controllers than other methods.

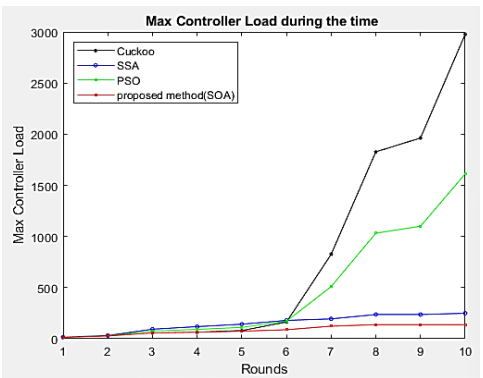


Fig. 8. The maximum controller load in Myren topology

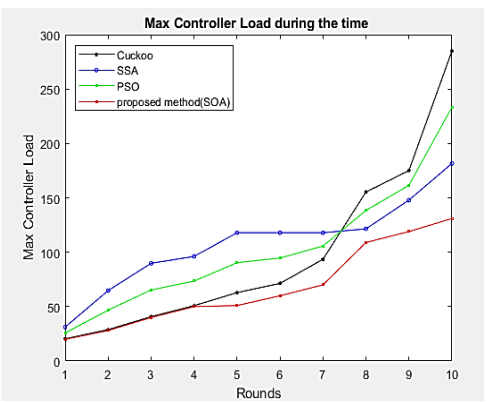


Fig. 9. The maximum controller load in Cogent topology

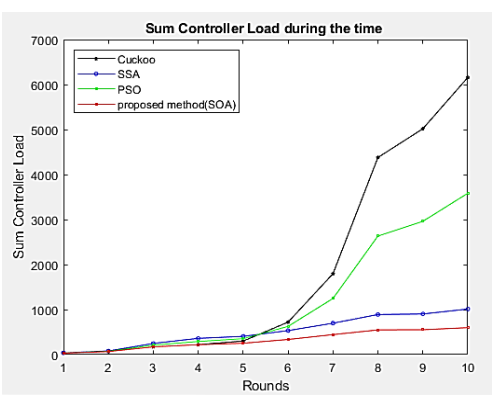


Fig. 10. The sum of the controller loads in Myren topology

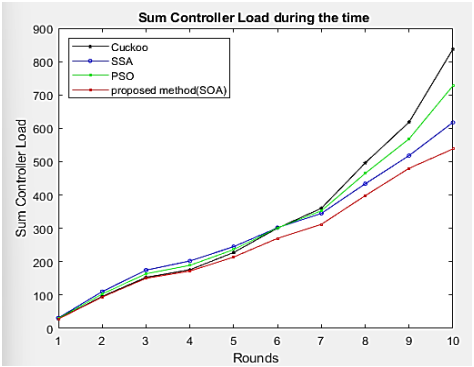


Fig. 11. The sum of the controller loads in Cogent topology

REFERENCES

[1] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, Proc. IEEE., vol. 103, no. 1, 2015.

[2] Masdari, M., Sangar, A. B., & Majidzadeh, K., "A Hybrid Multi-objective Algorithm for Imbalanced Controller Placement in Software-Defined Networks.," Journal of Network and Systems Management, vol. 30, no. 3, pp. 1-54, 2022.

[3] Jafarian, T., Masdari, M., Ghaffari, A., & Majidzadeh, K. , "Security anomaly detection in software-defined networking based on a prediction technique," International Journal of Communication Systems, vol. 33, no. 14, 2020.

[4] Jafarian, T., Masdari, M., Ghaffari, A., & Majidzadeh, K., "SADM-SDNC: security anomaly detection and mitigation in software-defined networking using C-support vector classification," Computing, vol. 103, no. 4, pp. 641-673, 2021.

[5] J. Liu, Z. Jiang, N. Kato, O. Akashi, and A. Takahara, "Reliability evaluation for NFV deployment of future mobile broadband networks," IEEE Wireless Commun., vol. 23, no. 3, pp. 90-96, 2016.

[6] JIE LU , ZHEN ZHANG, TAO HU , PENG YI, AND JULONG LAN, "A Survey of Controller Placement Problem in Software-Defined Networking," IEEE Access, vol. 7, pp. 24290-24307, 2019.

[7] tohid jafarian, Mohammad Masdari, ali ghaffari, kambiz majidzadeh, "A Survey and Classification of the Security Anomaly Detection Mechanisms in Software Defined Networks," Cluster Computing, vol. 24, p. 1235–1253 , 2021.

[8] Yuqi Fan, Lunfei Wang, Xiaohui Yuan, "Controller placements for latency minimization of both primary and backup paths in SDNs," Computer Communications, vol. 163, pp. 35-50, 2020.

[9] Jafarian, T., Masdari, M., Ghaffari, A., & Majidzadeh, K. , "Security anomaly detection in software-defined networking based on a prediction technique. ," International Journal of Communication Systems, vol. 33, no. 14, 2020.

[10] Manoharan, G. Ramya · R., "Enhanced optimal placements of multi-controllers in SDN," Journal of Ambient Intelligence and Humanized Computing, 2020.

[11] EL Hocine Bouzidi, Abdelkader Outtagarts, Rami Langar, Raouf Boutaba, "Dynamic clustering of software defined network switches and controller," Computer Networks, vol. 207, 2022.

[12] Beheshte Sadeghi Sabzevari, Mohammad Haddad Zarif, and Seyed Kamal Hosseini Sani, "Event-Triggered Predictive Networked Control Systems with Network Imperfections and External Disturbance," International Journal of Industrial Electronics, Control and Optimization, vol. 5, no. 1, pp. 11-22, 2022.

[13] B. Heller, R. Sherwood, N. McKeown,, "The controller placement problem,," in Proceedings of the First Workshop on Hot Topics in Software Defined Networks, in: HotSDN '12, ACM, , New York, NY, USA, 2012.

[14] Gaurav Dhiman, Vijay Kumar, "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems," Knowledge-Based Systems, vol. 165, pp. 169-196, 2019.

[15] Hassan Ghaedi; Seyed Reza Kamel Tabbakh Farizani; Reza Gaemi, "A Novel Meta-heuristic Framework for Solving Power Theft Detection Problem: Cheetah Optimization Algorithm," International Journal of Industrial Electronics, Control and Optimization , vol. 5, no. 1, pp. 63-76, 2022.

[16] Muhammad Fahad, Farhan Aadil , Zahoor-ur- Rehman, Salabat Khan , PeerAzmat Shah , Khan Muhammad , Jaime Lloret , Haox, "Grey wolf optimization based clustering algorithm for vehicular ad-hoc networks," Computers and Electrical Engineering, pp. 1-18, 2018.

[17] Saman M. Almufti, Ridwan B. Marqas, Pawan Sh. Othman, Amira Bibo Sallow, "Single-based and Population-based Metaheuristics for Solving NP-hard problems," Iraqi Journal of Science, vol. 62, no. 5, pp. 1710-1720, 2021.

[18] B. Heller, R. Sherwood, N. McKeown,, "The controller placement problem," in in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, in: HotSDN ', ACM, New York, NY, USA, 2012.

[19] K.S. Sahoo, S. Sahoo , A. Sarkar , B. Sahoo , R. Dash, "On the placement of controllers for designing a wide area software defined networks," in Proc. IEEE Region 10 Conference (TENCON), 2017.

[20] T. Zhang , A. Bianco , P. Giaccone, " The role of inter-controller traffic in sdn controllers placement," in Proc. IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 87–92 , 2016 .

[21] T. Zhang , P. Giaccone , A. Bianco , S.D. Domenico, "The role of the inter-controller consensus in the placement of distributed sdn controllers," Comput. Commun. , vol. 113, no. Supplement C, pp. 1-13, 2017.

[22] A. Dixit, F. Hao, S. Mukherjee, T.V. Lakshman, R.R. Kompella,, "ElastiCon: An Elastic Distributed SDN Controller," IEEE Comp Soc Tech Comm Comp Architecture, pp. 17-24, 2014.

[23] Ze Yang , Kwan L. Yeung, "MinimumWeight Controller Tree Design in SDN," Computer Networks, 2019.

[24] Dorabella Santos; Teresa Gomes; David Tipper, "SDN

Controller Placement With Availability Upgrade Under Delay and Geodiversity Constraints," IEEE Transactions on Network and Service Management , pp. 301 - 314, 2021.

[25] V. Ahmadi, M. Khorramizadeh ,, "An adaptive heuristic for multi-objective controller placement in software-defined networks," Comput. Electr. Eng., vol. 66, pp. 204-228, 2018.

[26] Soufian Toufga, Slim Abdellatif , Hamza Tarik Assouane, Philippe Owezarski, Thierry Villemur, "Towards Dynamic Controller Placement in Software Defined Vehicular Networks," Sensor, vol. 20, 2020.

[27] Hailan Kuang; Yiwen Qiu; Ruifang Li; Xinhua Liu, "A Hierarchical K-Means Algorithm for Controller Placement in SDN-Based WAN Architecture," in 2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Changsha, China, 2018.

[28] Guodong Wang; Yanxiao Zhao; Jun Huang; Qiang Duan; Jun Li, "A K-means-based network partition algorithm for controller placement in software defined network," in 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 2016.

[29] C. Gao, H. Wang , F. Zhu , L. Zhai , S. Yi, "A particle swarm optimization algorithm for controller placement problem in software defined network,," Cham, Springer International Publishing, p. 44–54, 2015.

[30] S. Liu, H. Wang , S. Yi , F. Zhu ,, "Ncpso: a solution of the controller placement problem in software defined networks,," in G. Wang, A. Zomaya, G. Martinez K. Li (Eds.), Algorithms and Architectures for Parallel Processing,, Cham, Springer International Publishing, Cham, pp. 213–225, 2015.

[31] X. X. Feixiang Li, "A Discrete Cuckoo Search Algorithm for the Controller Placement Problem in Software Defined Networks 2018 IEEE 9th Annual Information Technology,," in Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2018.

[32] Abdelhamied A. Ateya , Ammar Muthanna , Anastasia Vybornova , Abeer D. Algarni, "Chaotic salp swarm algorithm for SDN multi-controller networks," Engineering Science and Technology, an International Journal, vol. 22, pp. 1001-1012, 2019.

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

[34] Nasrin Firouz, Mohammad Masdari, Amin Babazadeh Sangar, Kambiz Majidzadeh, "A novel controller placement algorithm based on network portioning concept and a hybrid discrete optimization algorithm for multicontroller software-defined networks," Cluster Computing, 2021.

[35] Abeer A. Z. Ibrahim; Fazirulhisyam Hashim; Aduwati Sali; Nor K. Noordin; Saber M. E. Fa, "A Multi-Objective Routing Mechanism for Energy Management Optimization in SDN Multi-Control Architecture," IEEE Access, vol. 10, pp. 20312 - 20327, 2022.

[36] Xin Cui, Xiaohui Gao, Yan Ma, "An Optimized Controller Placement Algorithm in 5G Based on SDN," in 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 2020.

[37] Neamah S. Radam , Sufyan T. Faraj Al-Janabi and Khalid Sh. Jasim, "Multi-Controllers Placement Optimization in SDN by the Hybrid HSA-PSO Algorithm," Computers , vol. 11, no. 7, 2022.

[38] James F. Kurose, Keith W. Ross, in COMPUTER NETWORKING A Top-Down Approach, 8th ed., Pearson, , pp. 35-46, 2021.

[39] Marco Cello; Yang Xu; Anwar Walid; Gordon Wilfong; H. Jonathan Chao; Mario Marchese, "BalCon: A Distributed Elastic SDN Control via Efficient Switch Migration," in IEEE International Conference on Cloud Engineering (IC2E), Vancouver, BC, Canada, 2017 .

[40] J.C. Bezdek, R. Ehrlich, W. Full, "FCM: the fuzzy c-means clustering algorithm," Comput Geosci, vol. 10, no. 2, pp. 191-203, 1984.

[41] Gaurav Dhiman, Vijay Kumar, "Spotted hyena optimizer for solving complex and non-linear constrained engineering problems Harmony Search and Nature Inspired Optimization Algorithms," Springer, pp. 857-867, 2019.

[42] Seyedali Mirjalili, Seyed Mohammad Mirjalili, Andrew Lewis, "Grey wolf optimizer," Adv. Eng. Softw. , vol. 69, pp. 46-61, 2014.

[43] James Kennedy, Russell C. Eberhart,, "Particle swarm optimization," in Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948., 1995.

[44] Seyedali Mirjalili, "Moth-flame optimization algorithm: A novel natureinspired heuristic paradigm," Knowl.-Based Syst. , vol. 89, p. 228–249, 2015.

[45] Eric Bonabeau, Marco Dorigo, Guy Theraulaz, , "Swarm Intelligence: From Natural to Artificial Systems," in Oxford University Press, Inc., New York, NY, USA, , 1999.

[46] Choi, C. and J.-J. Lee, "Chaotic local search algorithm," Artificial Life and Robotics, vol. 2, no. 1, pp. 41-47, 1998.

[47] "Internet Topology Zoo," the University of Adelaide, [Online]. Available: http://www.topology-zoo.org/index.html. [Accessed 5 March 2022].



Maedeh Abedini Bagha received the B.Sc. degree in computer engineering software from Islamic Azad University, Zahedan, Iran, in 2007 and M.Sc. degree in computer engineering software from the Science and Research Branch of Islamic Azad University, Tabriz, Iran, in 2013. She is currently a Ph.D. candidate in computer software at the Islamic Azad University, Urmia Branch. Her research interests are in Distributed Systems, Optimization, and Metaheuristic algorithms.



Kambiz majidzadeh was born in Urmia, Iran, in 1980. He received the B.Sc. degree in software engineering from the Islamic Azad University of Khoy, Khoy, Iran, in 2002, and the M.Sc. degree in computer networking and the Ph.D. degree in information technology from Baku State University (BSU), Baku, Azerbayijan in 2005 and 2009, respectively. His research

interests are very large-scale integration design and computer networking.



Mohammad Masdari received his B.Tech. degree in Computer Software Engineering from Islamic Azad University, Qazvin Branch, Iran, in 2001, and M.Tech degree in Computer Software Engineering from Islamic Azad University, South Tehran Branch, Tehran, Iran, in 2003. He received his Ph.D. degree in Computer Software Engineering from Islamic Azad University, Science and Research Branch, Tehran, Iran, in 2014. Since 2003, he worked as a faculty member of Islamic Azad University, Urmia Branch, Iran. Presently he is an Assistant Professor in the Department of Computer Engineering of Islamic Azad University, Urmia Branch, Iran. His research interests include Distributed Systems and Network Security.



Yousef Farhang was born Khoy, Iran, in 1978. He received the B.S. degree in software engineering in 2001 from Islamic Azad University, Khoy, Iran, and his M.S. degree in software engineering in 2005 from Islamic Azad University, Arak, Iran. He received the Ph.D. degree in computer science in 2016 in Universiti Teknologi Malaysia (UTM). He was with Islamic Azad University, Khoy branch, as a researcher and lecturer from 2003 to now. His research interests include data mining, clustering algorithm, image clustering and machine learning algorithms.