



# A novel controller placement algorithm based on network portioning concept and a hybrid discrete optimization algorithm for multi-controller software-defined networks

Nasrin Firouz<sup>1</sup> · Mohammad Masdari<sup>1</sup> · Amin Babazadeh Sangar<sup>1</sup> · Kambiz Majidzadeh<sup>1</sup>

Received: 22 September 2020 / Revised: 15 January 2021 / Accepted: 14 March 2021 / Published online: 11 April 2021  
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

## Abstract

Software defined network (SDN) has shown significant advantages in numerous real-life aspects with separating the control plane from the data plane that provides programmable management for networks. However, with the increase in the network size, a single controller of SDN imposes considerable limitations on various features. Therefore, in networks with immense scalability, multiple controllers are essential. Specifying the optimal number of controllers and their deployment place is known as the controller placement problem (CPP), which affects the network's performance. In the present paper, a novel controller placement algorithm has been introduced using the advantages of nature-inspired optimization algorithms and network portioning. Firstly, the Manta Ray Foraging Optimization (MRFO) and Salp Swarm Algorithm (SSA) have been discretized to solve CPP. Three new operators comprising a two-point swap, random insert, and half points crossover operators were introduced to discretize the algorithms. Afterward, the resulting discrete MRFO and SSA algorithms were hybridized in a promoting manner. Next, the proposed discrete algorithm has been evaluated on six well-known software-defined networks with a different number of controllers. In addition, the networks have been chosen from various sizes to evaluate the scalability of the proposed algorithm. The proposed algorithm has been compared with several other state-of-the-art algorithms regarding network propagation delay and convergence rate in experiments. The findings indicated the effectiveness of the contributions and the superiority of the proposed algorithm over the competitor algorithms.

**Keywords** Software defined network · Controller placement · Delay · Network portioning · Discrete optimization

## 1 Introduction

In the last few years, following the rapid progression of hardware and software, Internet users significantly increased [1]. As a result of this leap, the amount of data traffics remarkably increased, which requires the

management, controlling, and upgrading of the foundational and infrastructure systems to fulfill the Internet's current demands [2–4]. Software-defined networks (SDN) is among the most popular technologies, transforming the traditional networking infrastructure into more adaptable, agile, flexible, and manageable network topologies [5, 6].

A software-defined network is an attention-grabbing network paradigm that decouples the control plane from the data plane in network switches [7, 8]. In this framework, control procedures are disconnected from the data plane and address to the external device, named controller [9, 10]. Therefore, developers are enabled to propose new algorithms on the SDN controller, which could manage the network and change its performance [11]. Figure 1 represents the three planes of SDN architecture, namely management, control, and data planes.

Switches exist in the data plane controllers, and management defines the plane policies and rules [12]. The

---

✉ Mohammad Masdari  
M.Masdari@Iaurmia.ac.ir

Nasrin Firouz  
n.firouz@iaurmia.ac.ir

Amin Babazadeh Sangar  
bsamin2@liveutm.onmicrosoft.com

Kambiz Majidzadeh  
K.majidzadeh@iaurmia.ac.ir

<sup>1</sup> Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran

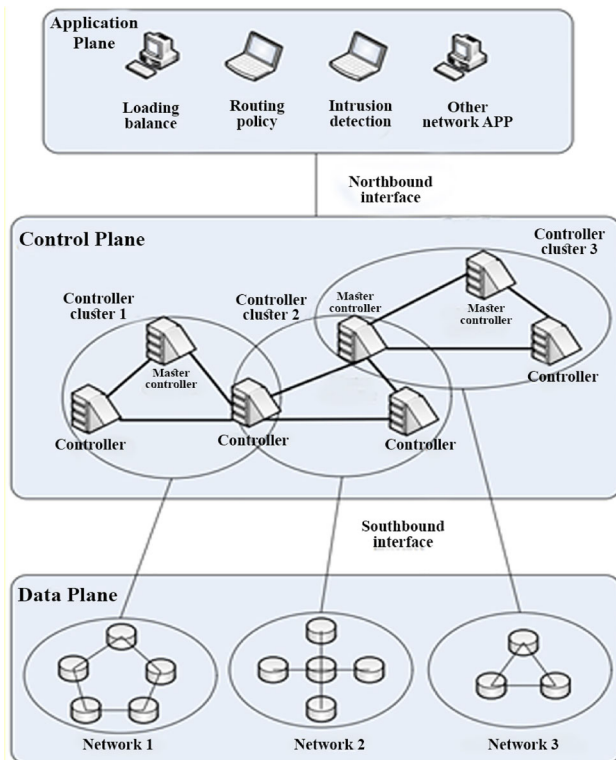


Fig. 1 The architecture of software-defined networks [24]

controller controls and manages the traffic of the network [8]. The switches' data in the network is sent to the corresponding controller for further switching of the processing [13]. The controllers in the software-defined networks manage and control the available switches' data streams by installing new flow rules and policies on them [14, 15]. However, especially in large-scale networks, a single controller may not be possible for a trade-off among the factors.

Furthermore, the performance of a single controller of SDN has many limitations. Consequently, using multiple controllers is a promising strategy to alleviate many factors such as reliability, resiliency, scalability, availability, minimum latency, and load balancing in a large-scale network [16–19]. One of the particularly vital issues in the SDN framework with multiple controllers is the controller placement problem (CPP) [20]. The controller placement problem involves two fields: the optimal number of controllers to be deployed in SDN, and these controllers' locations [21]. These characteristics affect the functionality of the network. The controller placement problem has been defined as an NP-hard problem [22]. It shows the number of required controllers to address the controllers' traffic loads and their location in the SDN cost-effectively. According to a series of factors, each switch would be linked to one of the SDN controllers [23].

Nature-inspired meta-heuristic algorithms are the utmost innovative and efficient approaches in solving complex real-world problems [25–28]. These algorithms have been applied to various problems and achieved promising results. Equilibrium Optimizer [29], Black Widow Optimization Algorithm [30], Seagull Optimization Algorithm [31], Deer Hunting Optimization Algorithm [32], Shuffled Shepherd Optimization [33], and Barnacles Mating Optimizer [34] are instances of the newly-developed meta-heuristic optimization algorithms. In the scope of software-defined networks, these algorithms have been utilized in numerous aspects embracing security, load balancing, routing, switch migration, etc. [15, 35–41]. Various solutions have been proposed for the controller placement problem based on optimization algorithms described in Sect. 2.

In the present paper, a novel meta-heuristic-based algorithm was proposed for controller placement in a software-defined network, which could provide dynamic load balancing and latency. In the proposed controller placement algorithm, the propagation latency between controllers and switches, dynamic traffic load, and linking switches to the best suitable controller were considered. The propagation latency in the SDN network was a critical factor since it affects network proficiency, especially in a large-scale network.

The critical contributions of this paper are as follows:

- The MRFO was innovatively discretized.
- The SSA was solitarily discretized.
- The discrete SSA and MRFO algorithms were hybridized using a multi-swarming concept.
- The greedy selection mechanism was employed in the resulting discrete hybrid algorithm.
- The proposed algorithm was applied to the CCP problem.
- Dijkstra's algorithm was embedded in the proposed algorithm to find the shortest routes between network switches.
- The Haversine formula was utilized to calculate the distance between the switches.
- The proposed controller placement algorithm was assessed on six real-world software-defined networks.
- The results of the proposed algorithm were compared with the state-of-the-art meta-heuristic-based algorithms.

The rest of the present paper is organized as follows: Sect. 2 is devoted to the related works. In Sect. 3, the background algorithms comprising SSA, MRFO, and Dijkstra's shortest path algorithms are described. In Sect. 4, the proposed controllers' placement algorithm is provided in detail for solving the CPP. Section 5 presents the extensive conducted experiments to evaluate the

performance of the proposed algorithm. Section 6 gives the conclusions and future perspectives.

## 2 Related work

Extensive amounts of effort have been dedicated to the controller placement in software-defined networks, summarized in Table 1. Some of the studies have been presented in this section. In [42], Ateya et al. proposed a meta-heuristic algorithm based on the chaotic Salp Swarm Optimization Algorithm (SSOA) for improving the optimizer performance. It dynamically determines the optimal number of controllers and the optimal paths between switches and the determined controllers in the large scale SDNs. This scheme was compared in various scenarios with other linear and metaheuristic algorithms for several real topologies obtained from the zoo topology. Their proposed method has brought improvements in terms of network latency and reliability. However, various network metrics such as latency, deployment cost, and system utilization have been considered in the present method, which requires more powerful hardware and higher running time. Additionally, to calculate the deployment cost, there was a requirement for more complex datasets and multiple

assumptions (which may lose their accuracy over time). Therefore, the computational complexity was high, and the precision could be conditional.

In [43], Gao et al. introduced a new algorithm for the controller placement problem in SDNs. The introduced algorithm considered controllers with capacity limitation, latency between controllers, and the latency between switches and controllers. In this scheme, a meta-heuristic algorithm was proposed based on the PSO algorithm for solving the problem, and the global latency is defined. Experiments showed that although the proposed algorithm minimized propagation latency, the controllers' Fixative traffic load was ignored. Besides, in the present paper, the problem and the discretizing approach was not apparent. Additionally, their proposed method was presented for the capacitated SDN networks.

In the controller placement approach provided by Hu et al. in [44], the focus was on defining the controllers' place for a distributed control plane to save SDN's energy. That is formulated as a binary integer program (BIP). In this model, the network's energy consumption that serves for the control traffic was reduced by a load of controllers and delay of control paths. Based on the high complexity of the BIP in wide-scale networks, the proposed IGCPA was an improved genetic controller placement algorithm, which

**Table 1** State-of-the-art of the controller placement in software-defined networks

	Scheme	Criteria	Algorithm	Networks	Year	Simulator
1	[42]	Utility of controllers	Salp swarm algorithm	ARPANET, MREN, GetNet, Sprint, NFS, Claranet, IBM, Oxford, FCCN, and AGIS	2019	MATLAB
2	[43]	Propagation latency	Particle swarm optimization	Randomly generated networks	2015	VS (C++)
3	[44]	Energy consumption	Genetic algorithm	Abilene, Janos-us, Pioro, and Zib	2016	MATLAB
4	[45]	Latency-workload	Hierarchical distributed architecture	Campus map of Nanyang Technological University	2018	NS-3
5	[46]	Latency	Verna-based optimization	Abilene, Savvis, Biznet, and Internet2 OS3E	2020	MATLAB-Python
6	[47]	Latency	K-means	Internet2 OS3E and ChinaNet	2017	MATLAB
7	[48]	load balancing	Rounding-based multi-area routing	VL2 and fat-tree	2018	Mininet
8	[49]	Latency-link failure	Grey wolf optimization	AT&t, BTNA, and Sprint	2020	Python
9	[50]	Latency	Hybrid particle swarm optimization-simulated annealing	Surfnet, Forthnet, and TataNId	2020	Python
10	[51]	Delay-load balancing	Firefly algorithm	A randomly generated network	2019	MATLAB
11	[52]	Latency	Particle swarm optimization and Firefly algorithm	TataNId, Fouthnet, Deutsche, Geant, and RNP	2018	MATLAB
12	[53]	Latency-load balancing	NSGA-II	NTT, Bisc, Garnet, Xspedius, Cernet, Litnet, Ntelos, Bell Canada, and Bellsouth	2018	MATLAB
13	[54]	Distance	Cuckoo search	Four randomly generated networks	2020	Java

is used to find an efficient sub-optimal solution for the power-aware CPP. The results indicated that IGCPA was close to that of the optimal solution. However, they did not consider the energy-aware issue on the data forwarding plane for a complete energy-aware SDN. Moreover, their method is still extensively complicated on large-scale networks.

In addition, in [45], Liyanage et al. presented a new hierarchically distributed Software-Defined Vehicular Network (HD-SDVN) framework with a controller placement problem to optimize the placement of the controllers regarding the delaying limitation in VANET. The framework divides the control plane into the top level, deployed on the Internet, and the bottom level, deployed at the Road Side Unit (RSU) level. Incorporating the existing SDVN framework and conventional Vehicular Ad-hoc Network (VANET), they indicated that their controller placement scheme could cause a lower latency than other possible controller placement schemes. They presented a scheme that obtained fair results concerning latency and workload in regional networks. Besides, various delays (transmission delay, queuing delay, contention delay, processing delay, propagation delay) are calculated in the presented paper, which is not necessarily required to be calculated all-in-one. Therefore, it could be deduced that the presented algorithm consumes a significant amount of time on calculating these delays.

Furthermore, in [46], Singh et al. have addressed controller placement in SDN by developing a new optimization algorithm named Varna-Based Optimization (VBO) to solve the capacitated controller placement problem. They analyzed the solution based on the optimization and clustering concept for CPP. Their simulation results indicated that the solution based on the optimization had better results than the clustering one, which minimized SDN's overall average latency. Its convergence rate was better than PSO, TLBO, and Jaya algorithms. Also, VBO classified particles just in two classes, where each class had a specific task and could be dynamically enhanced by changing the number of classes during generations. The proposed method had various superiorities over previous methods; however, it was not usable for all SDN networks and was provided for capacitated networks.

Besides, an approach is proposed by Wang et al. in [47] to minimize latency between controllers and their switches. They estimated qualitatively by considering the total latency between switches and controllers. They also proposed a partition method based on clustering to partition a network into subnetworks to decrease the end-to-end latency and the queuing latency of controllers. This algorithm ensured that each partition could minimize the maximum end-to-end delay between switches and controllers. Multiple controllers are proposed to be established

into each subnetwork to reduce queuing latency by extreme packet requests from switches. A series of simulations were conducted using two real-world topologies from the Zoo Topology to analyze the proposed algorithm's performance. The simulation results showed that the overall latency decreased by multiple controllers deployed in subnetworks compared with K-center and K-means. The proposed algorithm was applied to only two networks, making it irritable to evaluate the algorithm's performance accurately.

Moreover, in [48], Wang et al. presented a routing-based algorithm to solve the controllers' load balancing problem in a software-defined network. They considered an effective mechanism for network status maintenance between distributed controllers to increase their algorithm's performance and analyzing the estimation performance. The simulation results showed that although their proposed algorithm could not decrease link load balancing performance, compared with the multi-thread successor of the NOX (NOX-MT) solution, it could decrease the maximum controller response time by 70% [55]. The proposed algorithm could find proper controller placement to decrease the response time and balance the network load; however, the present paper's main focus was on the routing concept. Therefore, the experiments were mainly conducted concerning the link load, and little attention has been paid to other network parameters.

Furthermore, Singh et al. developed a novel optimization algorithm called varna-based optimization (VBO) in [46] and used it to solve the SDN controller placement problem. The proposed algorithm's main goal was to minimize average network latency, defined as the VBO algorithm's objective. They evaluated the proposed algorithm on Internet2 and Savvis SDN networks, and the experimental results showed the superiority of the proposed algorithm. The proposed algorithm minimized the average network latency and had low complexity; however, the experiments were insubstantial. Additionally, the algorithm was applied to only two networks and did not compare with state-of-the-art algorithms accurately. Besides, the discretizing method and operators were not transparent, and the complexity was not discussed.

Moreover, in [11], Moradi et al. proposed a new controller placement algorithm based on iterated local search for SDN networks. In the proposed algorithm, network deployment costs, including installing the controllers, connecting switches to the controllers, and connecting controllers to other controllers, were considered objective. Also, the local search was used to find better solutions by searching the neighbors. The proposed algorithm was compared with CPLEX and SCIP solvers. Furthermore, the performance of the algorithms was evaluated on several random networks on a  $20 \times 20$  grid. The results



demonstrated that the proposed algorithm was computationally efficient. In the proposed algorithm, there was a vacancy for the meta-heuristic algorithm. Furthermore, the proposed method was only tested on small networks and compared with a few algorithms.

Besides, in [21], Schütz and Martins suggested an approach to find optimal places for SDN controllers. The suggested algorithm constrained propagation latency and controller capacity. Also, it able to find the minimum number of controllers and their locations. Once the controller's optimal number and location were found, the suggested algorithm connected to a controller and balanced the controllers' load. The suggested algorithm was evaluated on fifteen SDN network and balanced the load better than competitor algorithms. Besides, it considered crucial network parameters such as latency, capacity, and load balance. It used a heuristic approach to find optimal places; however, there was a lack of investigation into the impact of using meta-heuristic algorithms in this study. Since further improvements could be achieved by using meta-heuristic algorithms.

Furthermore, in [52], Sahoo et al. provided a novel controller placement algorithm for SDN networks. The provided algorithm supplied a seamless backup technique against single link failure, minimizing communication delay using survivability. The particle swarm optimization and Firefly algorithm were used in the provided algorithm to achieve the targets mentioned above. Also, latency between controllers, latency between switches and controllers, and multi-path connectivity between the switches and controllers were taken into accounts in the provided algorithm. The performance of the provided algorithms was evaluated on a set of publicly SDN networks. The experimental results showed that the provided algorithm improved the control path's survivability and the network's performance effectively. The proposed algorithm used a hybrid meta-heuristic optimization algorithm to solve the CPP problem, bringing significant advantages. Furthermore, the proposed method was evaluated by various SDN networks that tested the algorithm's performance more accurately. Nevertheless, the optimization algorithms used in this paper (PSO and FFA) were old ones that might not be efficient for today's problems. The use of newer optimization algorithms, newer improvement methods, newer combining approaches, etc., would increase the algorithm's efficiency. Moreover, another predicament of this paper was that the proposed algorithm was compared with a small number of algorithms, which could not correctly show the superiority of the proposed algorithm.

Additionally, Jalili et al. presented a novel controller placement algorithm using NSGA-II for large-scale software-defined networks [53]. In the presented algorithm, maximum and average latencies and load balancing on

controllers were considered objective functions. Several networks from the Topology Zoo evaluated the presented algorithm, and the obtained results demonstrated the superiority of the presented algorithm over the other similar algorithms. The presented multi-objective controller placement algorithm optimized the latency and balanced controllers' load using NSGA-II. The inverted generational distance (IGD) metric was also used to evaluate the Pareto Front obtained by the algorithms. Nonetheless, the NSGA-II algorithm was older, and newer multi-objective algorithms could be used instead. The Pareto Front obtained by the proposed algorithm and the competitor algorithms did not plot and compared efficiently. Moreover, although the proposed algorithm was applied to several SDN networks, it was compared with PSA and PSO algorithms, which did not accurately express the proposed algorithm's possible superiority.

Moreover, in [54], Tahmasebi et al. suggested a novel synchronization-aware controller placement algorithm for WSN networks to optimize network performance. The optimization process was done in the suggested algorithm using the cuckoo search algorithm, which is a meta-heuristic optimization algorithm. The performance of the suggested algorithm was evaluated by comparing it with simulated annealing and quantum annealing algorithms. The results of the comparisons demonstrated that the suggested algorithm outperformed competitor algorithms. In the proposed algorithm, various network metrics were considered. The method's overall complexity and reliability were good; however, it was applied to the WSN networks. The proposed algorithm could apply to SDN networks with some modifications, and the newer meta-heuristic algorithm could be employed instead of the Cuckoo search algorithm.

Additionally, Nasiri and Derakhshan investigated the SDN networks and presented a novel greedy method to discover an optimal way to embed offline VNE into OpenFlow networks and manage communications between the virtual controllers and switches [56]. In the presented paper, various metrics such as node capacity, link capacity, revenue, cost, latency, CPU, and bandwidth resources were considered. The performance of the presented method was investigated in different environments regarding revenue, cost, and delay.

In this paper's presented method, most-recently developed powerful algorithms (MRFO and SSA) were recruited, and the simplicity and low complexity were the authors' priorities. Furthermore, the presented paper provides a comprehensive approach for all kinds of SDN networks rather than a specific network range. Therefore, the proposed algorithm was applied to various SDN networks in terms of its comprehensiveness and to evaluate the algorithm's reliability. The networks were selected

from real-world SDN networks (Topology Zoo) with different sizes (small- and large-scales). Afterward, the proposed algorithm was compared with various neoteric meta-heuristic-based controller placement algorithms to demonstrate the algorithm’s superiority.

### 3 Research background

#### 3.1 Manta ray foraging optimization algorithm

Manta ray foraging optimization (MRFO) is a novel bio-inspired optimization algorithm inspired by manta rays’ intelligent behaviors in nature [57]. MRFO was developed to solve various optimization problems and was composed of three phases: chain foraging, cyclone foraging, and somersault foraging. These phases model the behaviors of the manta rays mathematically and are described as follows.

##### 3.1.1 Chain foraging

This phase of the algorithm models the observation and movement of the mana rays toward plankton. Manta Rays line up head-to-tail and form a foraging chain. The chain was modeled as below:

$$X_i^d(t + 1) = \begin{cases} X_i^d(t) + r.(X_{best}^d(t) - X_i^d(t)) + \alpha.(X_{best}^d(t) - X_i^d(t)) & i = 1 \\ X_i^d(t) + r.(X_{i-1}^d(t) - X_i^d(t)) + \alpha.(X_{best}^d(t) - X_i^d(t)) & i = 2 \dots N \end{cases} \tag{1}$$

where,  $X_i^d$  is the position of  $i$ th manta ray,  $t$  indicates time,  $d$  is the dimension,  $r$  is a vector with random numbers in  $[0, 1]$ ,  $X_{best}^d(t)$  is the plankton with a high concentration in time  $t$ , and  $\alpha$  is a weight coefficient which was calculated as follow:

$$\alpha = 2.r.\sqrt{|\log(r)|} \tag{2}$$

##### 3.1.2 Cyclone foraging

This phase models the long foraging chain of the manta rays formed to swim toward the food by a spiral. This foraging chain was similar to the foraging strategy of the Whale optimization algorithm [58]. However, in the cyclone foraging strategy, each manta ray swims toward the one in front of it using the manta rays’ spiral movements. A manta ray moves toward one of the fronts-lines and moves toward the food along a spiral path. Equation (3) models this behavior of the manta rays mathematically.

$$X_i^d(t + 1) = \begin{cases} X_{best}^d + r.(X_{best}^d(t) - X_i^d(t)) + \beta.(X_{best}^d(t) - X_i^d(t)) & i = 1 \\ X_{best}^d(t) + r.(X_{i-1}^d(t) - X_i^d(t)) + \beta.(X_{best}^d(t) - X_i^d(t)) & i = 2 \dots N \end{cases} \tag{3}$$

where  $T$  is the maximum number of iterations, and  $\beta$  is the weight coefficient that could be defined as below:

$$\beta = 2e^{r_1 \frac{T-t+1}{T}} * \sin(2\pi r_1) \tag{4}$$

where  $r_1$  in a random number in  $[0,1]$ .

This cyclone foraging strategy has excellent exploitation for a part of the search space that the plankton is in it, as the manta rays search the problem space concerning the food. The manta rays are forced to search for new positions far from the current best manta ray to increase the exploration. In MRFO, this is modeled by assigning a new random position in the problem search to the manta rays as their reference position. The following equation mathematically models this strategy:

$$X_i^d(t + 1) = \begin{cases} X_{rand}^d + r.(X_{rand}^d(t) - X_i^d(t)) + \beta.(X_{rand}^d(t) - X_i^d(t)) & i = 1 \\ X_{rand}^d(t) + r.(X_{i-1}^d(t) - X_i^d(t)) + \beta.(X_{rand}^d(t) - X_i^d(t)) & i = 2 \dots N \end{cases} \tag{5}$$

where  $X_{rand}$  in a random position in the search space and is produced as follow:

$$X_{rand}^d = lb^d + r.(ub^d - lb^d) \tag{6}$$

where  $lb$  and  $ub$  are the lower and upper bounds of the search space, respectively.

##### 3.1.3 Somersault foraging

In this phase, the best previously discovered individual was considered a pivot, and the manta rays update their position by swimming and turning around the pivot. Therefore, this phase increases the exploitation capability of the MRFO. Equation (7) models this behavior of the manta rays mathematically.

$$X_i^d(t + 1) = X_i^d(t) + S.(r_2.X_{best}^d(t) - r_3.X_i^d(t)) + \beta.(X_{rand}^d(t) - X_i^d(t))i = 1 \dots N \tag{7}$$

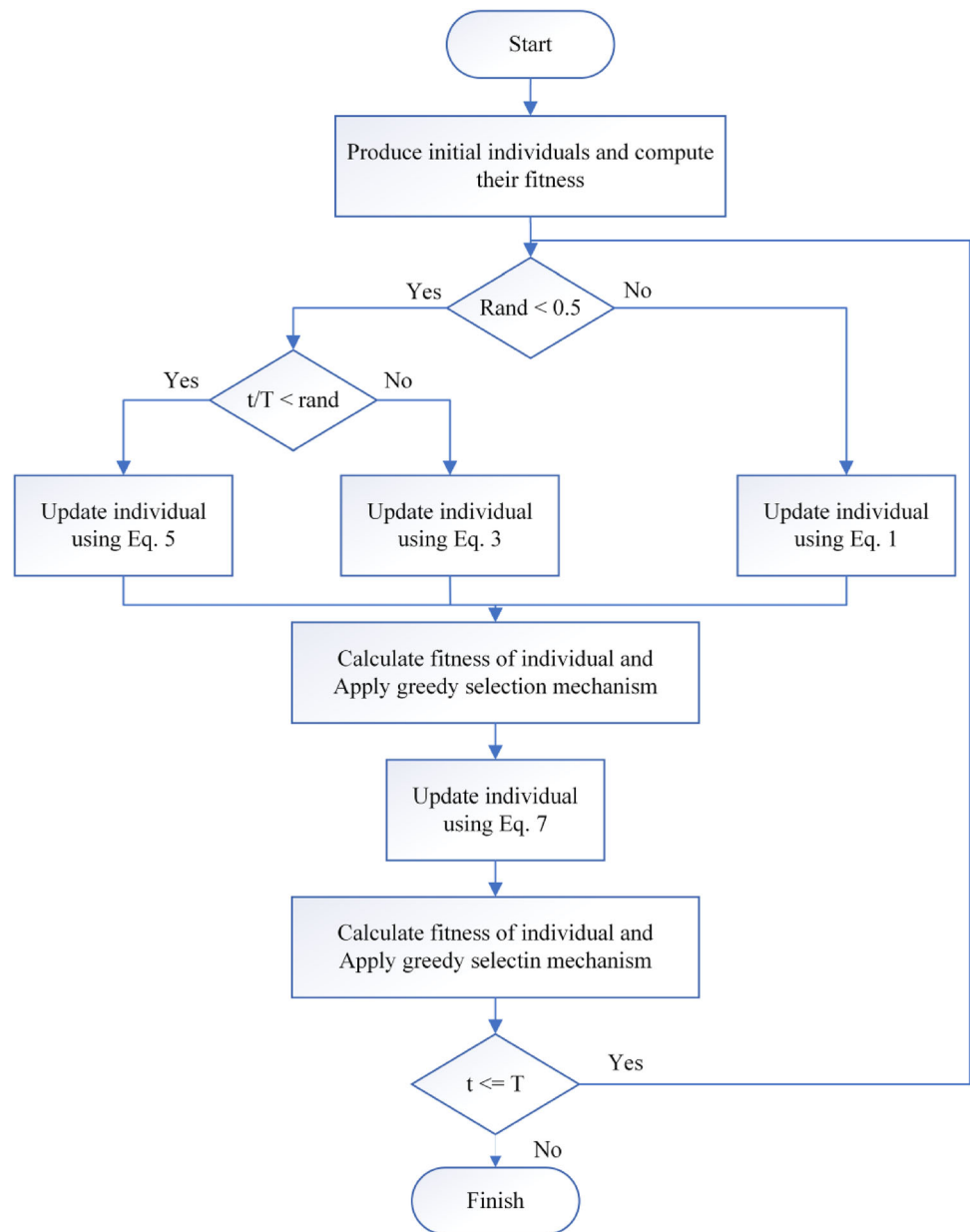
where  $S$  is the somersault factor, which decides the somersault range of manta rays,  $r_2$  and  $r_3$  are random numbers between  $[0, 1]$ .

The flowchart of MRFO is illustrated in Fig. 2.

### 3.2 SSA algorithm

Salp swarm algorithm (SSA) is a robust nature-inspired optimization algorithm developed by Mir Jalili et al. in [59]. SSA has very low complexity and is inspired by the Salp chain’s two foraging behaviors in nature. The salps are very similar to jellyfish and use water to move toward

**Fig. 2** Flowchart of the MRFO algorithm



food. In SSA, the salp swarm divides into two groups: leader and followers. The leader is at the front of the chain and guides the swarm. The remaining salps in the swarm were considered as followers. The leader in the swarm updates its position using Eq. (8).

$$X_1^j = \begin{cases} X_{best}^j + c_1(c_2.(ub^j - lb^j) + lb^j) & c_3 \geq 0 \\ X_{best}^j + c_1(c_2.(ub^j - lb^j) + lb^j) & c_3 < 0 \end{cases} \quad (8)$$

where  $X_1^j$  is the  $j$ th dimension of the leader,  $X_{best}$  in the position of the best solution that is found so far,  $c_2$  and  $c_3$  are random numbers in  $[0, 1]$ , and  $c_1$  balances exploration and exploitation and was calculated as follow:

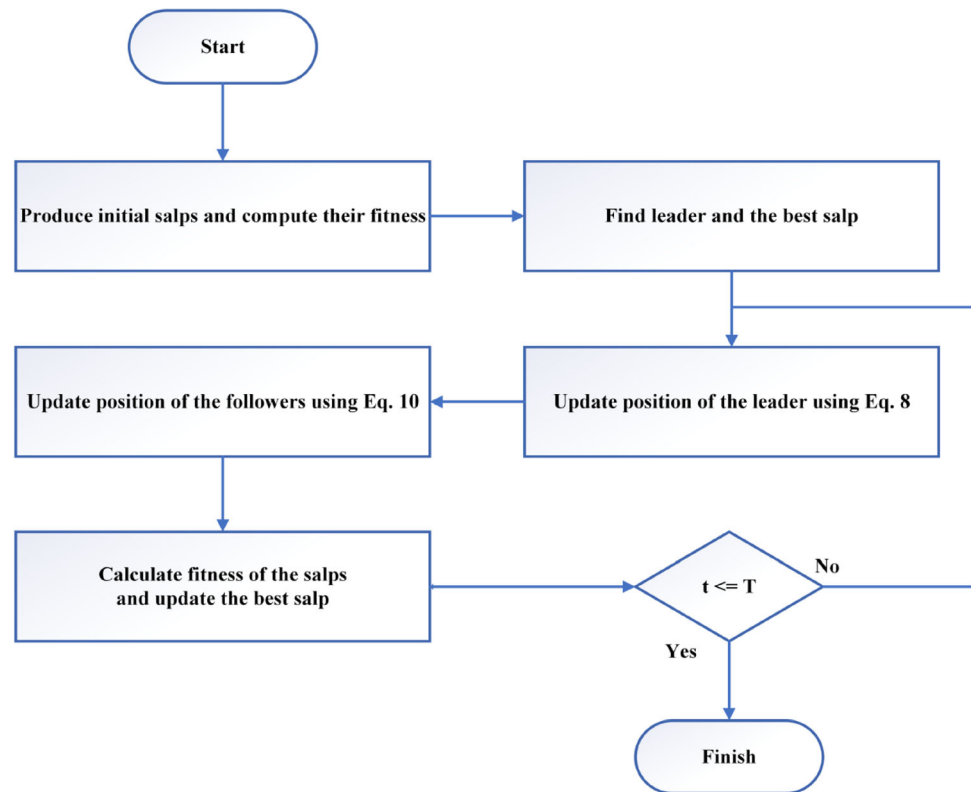
$$c_1 = 2e^{-\frac{t}{T}} \quad (9)$$

where  $t$  and  $T$  are the current and maximum number of iterations, the follower salps update their positions using Eq. 10.

$$X_i^j = \frac{1}{2}(X_i^j - X_{i-1}^j) \quad (10)$$

where  $X_i^j$  is the  $j$ th dimension of the  $i$ th salp and  $i > 1$ . Figure 3 shows the flowcharts of SSA.

**Fig. 3** Flowchart of the SSA algorithm



## 4 Proposed controller placement solution

As mentioned in previous sections, controller placement is one of the most critical issues in SDN networks. By selecting the appropriate number of controllers and their locations, the network delay could be significantly reduced, and the network load could be distributed. In the current paper, a new method was proposed for selecting the controllers' optimal location by using MRFO and SSA optimization algorithms, which were discretized in a novel way. The controller placement algorithm proposed in the present paper consists of several pieces described in detail in the following subsections.

### 4.1 Discrete MRFO algorithm

Even though the basic Manta ray foraging optimization algorithm is suitable for solving continuous problems, the controller placement problem is discrete. Therefore, in the present paper, MRFO was modified by two new operators to solve discrete problems. For this purpose, Eq. (1) was replaced by a two-point swap operator with  $X_i$  and  $X_{best}$  as ins inputs. It is worth mentioning that the order of the inputs is significant, and by exchanging the order of input solutions, the results will be changed significantly. The pseudocode of the two-point swap operator is presented in Fig. 4.

In Fig. 4,  $SP1$  and  $SP2$  are swap points that were selected randomly. Figure 5 illustrates the results of the two-point swap operator.

In Fig. 5, the values of  $SP1$  and  $SP2$  are 2 and 5, respectively. Besides, Eq. (3) was replaced with the two-point swap operator with  $X_{best}$  and  $X_i$  as its inputs. Furthermore, Eq. (5) was replaced by a random insert operator. The pseudocode of this new operator was presented in Fig. 6. The function of the random insert operator is illustrated in Fig. 7.

In Fig. 6,  $NC$  is the number of controllers,  $RN$  is a random integer number between  $[1NC]$ ,  $\overrightarrow{RP}$  is a vector with random numbers in  $[1NC]$ , and  $randperm$  generates a vector with  $RN$  random integer numbers in  $[1NP]$ .

In Fig. 7,  $NC$  is 6,  $RN$  is 3,  $\overrightarrow{RP}$  is  $[2, 3, 5]$ , and as it can be observed clearly in the figure, the points 2,3 and 5 are replaced with random indices. Additionally, Eq. (7) was replaced by Eq. (11).

$$X_i^j = \begin{cases} X_{best}^j & R < 1/3 \\ X_r^j & 1/3 < R < 2/3 \\ RI & 2/3 < R < 1 \end{cases} \quad (11)$$

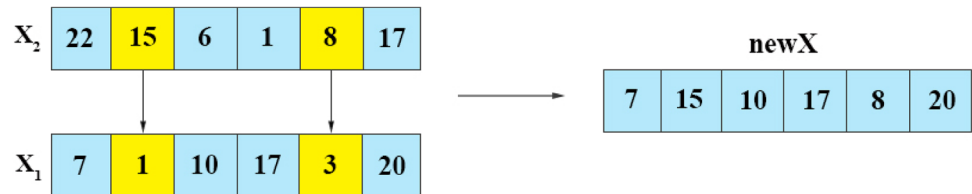
where  $R$  is a random number in  $[0, 1]$ ,  $X_r$  is a randomly chosen solution from the current swarm, and  $RI$  is a random index. Figure 8 depicts the pseudocode of the proposed discrete MRFO.

**Input:** two solutions ( $X_1, X_2$ )  
**Output:** a new solution ( $newX$ )

Select two points randomly from the solutions,  $SP1$ , and  $SP2$   
 $newX = X_1$   
 $newX^{SP1} = X_2^{SP1}$   
 $newX^{SP2} = X_2^{SP2}$   
 Return  $newX$

Fig. 4 Pseudocode of the two-point swap operator

Fig. 5 Two-point swap operator



**Input:** a solution ( $X_i$ ) and the number of controllers ( $NC$ )  
**Output:** a new solution ( $newX$ )

$RN = \text{floor}(rn * NC) + 1$       %  $rn$  is a random number in  $[0,1]$ , and  $NC$  is the number of controllers  
 $\overrightarrow{RP} = \text{randperm}(NC, RN)$       % Select  $RN$  points from the solution randomly  
 $newX = X_i$   
 for  $j = 1 : RN$   
      $newX^{RP(j)} = \text{floor}(\text{rand}(0,1) * NS) + 1$   
 end for  
 Return  $newX$

Fig. 6 Pseudocode of the random insert operator

Fig. 7 Random insert operator



### 4.2 Discrete salp swarm algorithm

Like the MRFO, the SSA algorithm is a continuous algorithm designed to solve continuous problems. This subsection presents the proposed discrete SSA algorithm. In the literature, discrete versions of the SSA algorithm have been proposed by scholars [42, 60, 61]; however, the presented discrete version in this paper is a novel method that has not been tapped before. For this purpose, Eq. (8) was replaced by Eq. (12).

$$X_i^j = \begin{cases} X_r^j & \text{if } c3 > 0.5 \\ X_{best}^j & \text{otherwise} \end{cases} \quad (12)$$

where  $c3$  is a random number in  $[0, 1]$ .

Moreover, Eq. (10) was replaced by the half points crossover operator. Figure 9 presents the pseudocode of the half points crossover operator.

The function of the half points crossover operator is shown in Fig. 10. In this figure,  $NC$  is 6,  $HP$  is 3,  $CP1$  is  $[1, 2, 5]$ , and  $CP2$  is  $[2, 4, 5]$ . The pseudocode of the proposed discrete SSA algorithm is provided in Fig. 11.

### 4.3 Dijkstra’s shortest path algorithm

In the proposed algorithm, Dijkstra’s shortest path algorithm was used to find the shortest path between switches in the SDN. The pseudocode of Dijkstra’s shortest path algorithm is provided in Fig. 12 [62].

It should also be mentioned that the position of SDN switches was expressed based on latitude and longitude. Therefore, a particular mechanism was needed to find the distance between the switches in kilometers. Various methods, such as cosine-haversine formula [63], Pythagorean theorem [64, 65], law of cosines [66], etc., were



```

Coef = rand(0,1)
for i = 1 to nSol
    if rand < 0.5
        if Coef > rand
            Xi = random_insert(Xi, NC)
        else
            Xi = two_point_swap(Xbest, Xi)
        end if
    else
        Xi = two_point_swap(Xi, Xbest)
    end if
end for

for i = 1 to nSol
    for j = 1 to NC
        R = rand(0,1)
        if R < 1/3
            Xij = Xbestj
        elseif R < 2/3
            Xij = Xrj
        else
            Xij = floor(rn * NS) + 1    % rn is a random number in [0,1], and NS is the number of switches
        end if
    end for
end for

```

**Fig. 8** Pseudocode of the proposed discrete Manta ray foraging optimization algorithm

**Input:** a solution ( $X_i$ ) and number of controllers (NC)

**Output:** a new solution

Generate a new random solution,  $X_r$

$newX = X_i$

$HP = \text{floor}(NC/2)$ ;

$CP1 = \text{randperm}(NC, HP)$

$CP2 = \text{randperm}(NC, HP)$

for  $j = 1 : HP$

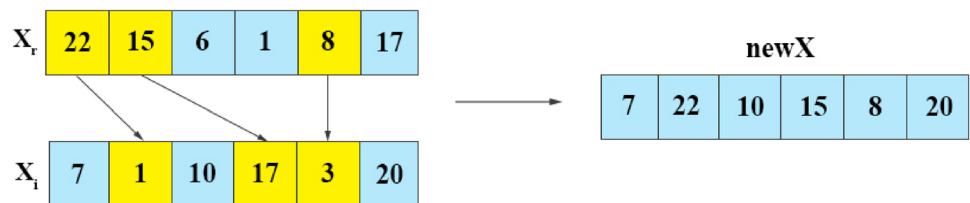
$newX^{CP1(j)} = X_r^{CP2(j)}$

end for

Return  $newX$

**Fig. 9** Pseudocode of the proposed half points crossover operator

**Fig. 10** Half points crossover operator



proposed to calculate the distance between two longitudes and latitudes based on kilometers. In the proposed algorithm, the cosine-haversine formula was used. In the cosine-haversine formula, the distance was calculated as follow:

$$d = 2 \cdot r \cdot \arcsin(\sqrt{\text{harv}}) \quad (13)$$

where the  $r$  is the earth's radius, which varies from 6356.752 km at the poles to 6378.137 km at Ecuador. Additionally,  $\text{harv}$  is the haversine function that was defined as below:

```

for i = 1 to nSol
    c2 = rand(0,1)
    if c2 < 0.5
        for j = 1 : NC
            c3 = rand(0,1)
            if c3 < 0.5
                Xij = Xrj
            else
                Xij = Xbestj
            end if
        end for
    else
        Xij = half_points_crossover(Xij, NC)
    end if
end for
    
```

Fig. 11 Pseudocode of the proposed discrete SSA algorithm

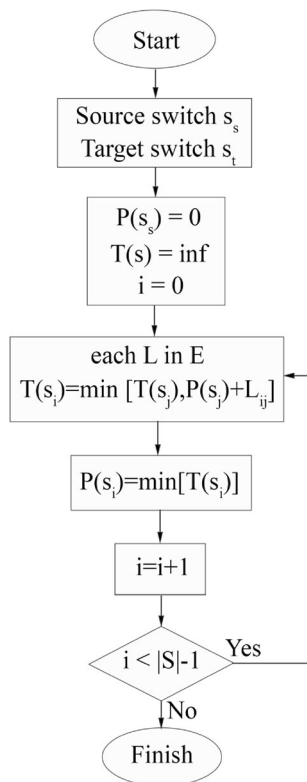


Fig. 12 Dijkstra's shortest path algorithm

$$harv = \sin(In_1)^2 + \cos(\varphi_1)\cos(\varphi_2)\sin(In_2)^2 \tag{14}$$

$$In_1 = \frac{\varphi_1 - \varphi_2}{2} \tag{15}$$

$$In_2 = \frac{\gamma_1 - \gamma_2}{2} \tag{16}$$

where  $\varphi_1$  and  $\gamma_2$  are latitude and longitude of the first switch,  $\varphi_2$  and  $\gamma_2$  are latitude and longitude of the second switch in radians, respectively. Also, Eq. (17) was used to convert longitude and latitude from degree to radian.

$$\begin{aligned}
 \text{latitude}(\text{radian}) &= \frac{\text{latitude}(\text{degree}) \times \pi}{180} \\
 \text{longitude}(\text{radian}) &= \frac{\text{longitude}(\text{degree}) \times \pi}{180}
 \end{aligned}
 \tag{17}$$

where,  $\pi = 3.1415$ .

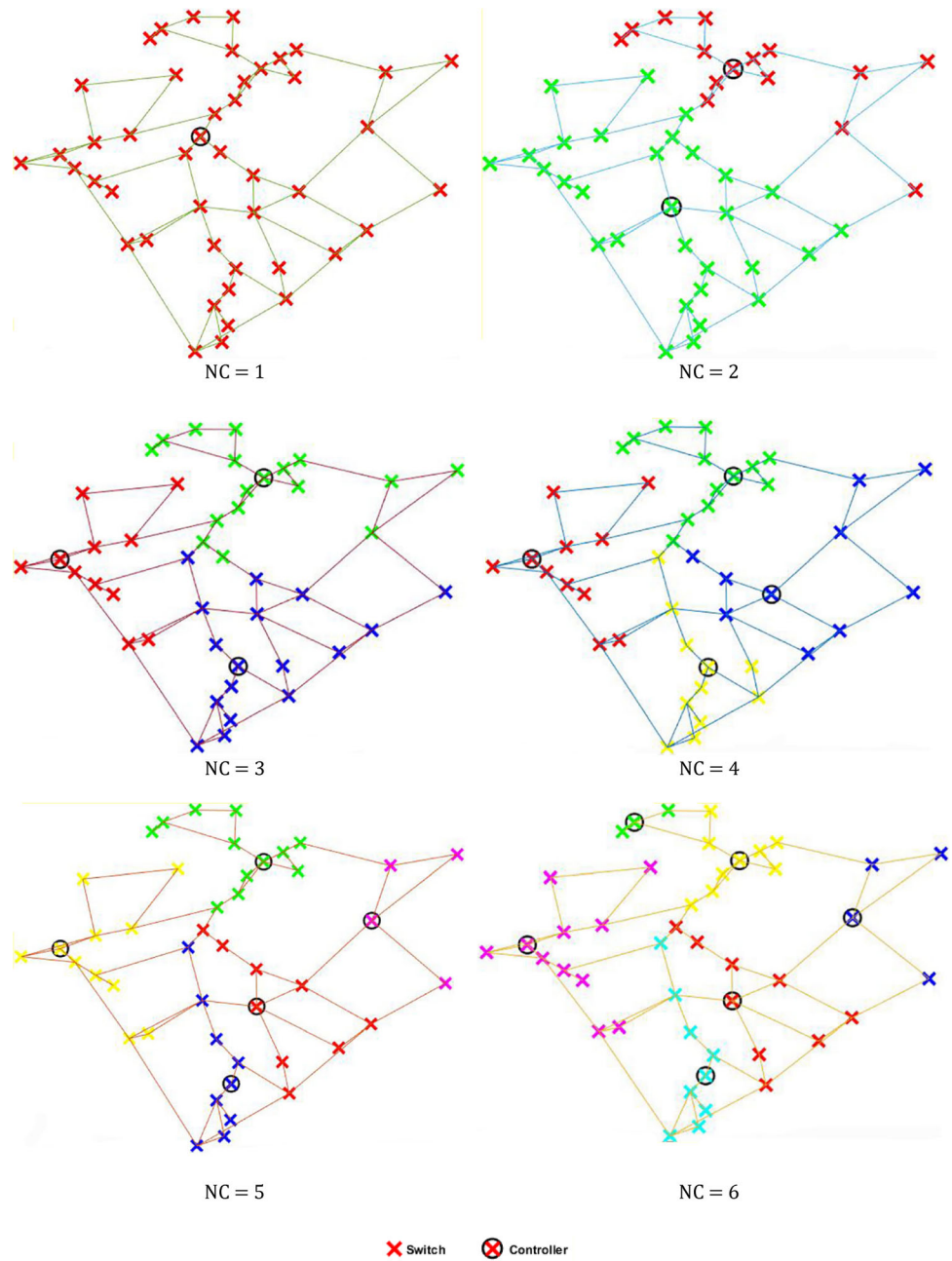
#### 4.4 Problem formulation

In the current paper, the controller placement problem was attentively scrutinized, and a novel method was proposed to reduce the overall delay of the software-defined networks (SDNs) by splitting it into several sections. In this case, the network is split into several sub-network, in which each sub-network has a controller. To mathematically model the network partition and controller placement problem, the following assumptions were taken into account.

- The location of all network switches was given and was consistent.
- The length, bandwidth, and traffic through the link were recognized.
- The numbers and locations of controllers could be changed.
- The communication links were given and were consistent.

The controller placement problem with a different number of sub-networks is demonstrated in Fig. 13. The SDN networks could be shown by a graph  $G = (S, E)$ , which  $S = \{s_1, s_2, \dots, s_n\}$  is the set of switches, and  $E$  is the set of communication links between switches. The  $S$  was formulated as:

**Fig. 13** Controller placement problem with different number of controllers



$$S = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ \vdots & \vdots \\ X_n & Y_n \end{bmatrix}_{n \times 2} \tag{18}$$

where the  $n$  is the number of switches. Besides, the set of links between switches could be illustrated by a  $NS \times NS$  matrix as below:

$$E = \begin{bmatrix} l_{1,1} & l_{1,2} & \dots & l_{1,n} \\ l_{2,1} & l_{2,2} & \dots & l_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ l_{n,1} & l_{n,2} & \dots & l_{n,n} \end{bmatrix}_{NS \times NS} \tag{19}$$

where  $NS$  is the number of switches in the network, and the elements of the matrix were defined as follows [67]:

$$l_{i,j} = \begin{cases} 1 & \text{if there is a link between } i\text{th and } j\text{th switches} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

15	24	3	10
----	----	---	----

Fig. 14 Encoding the controller selection process

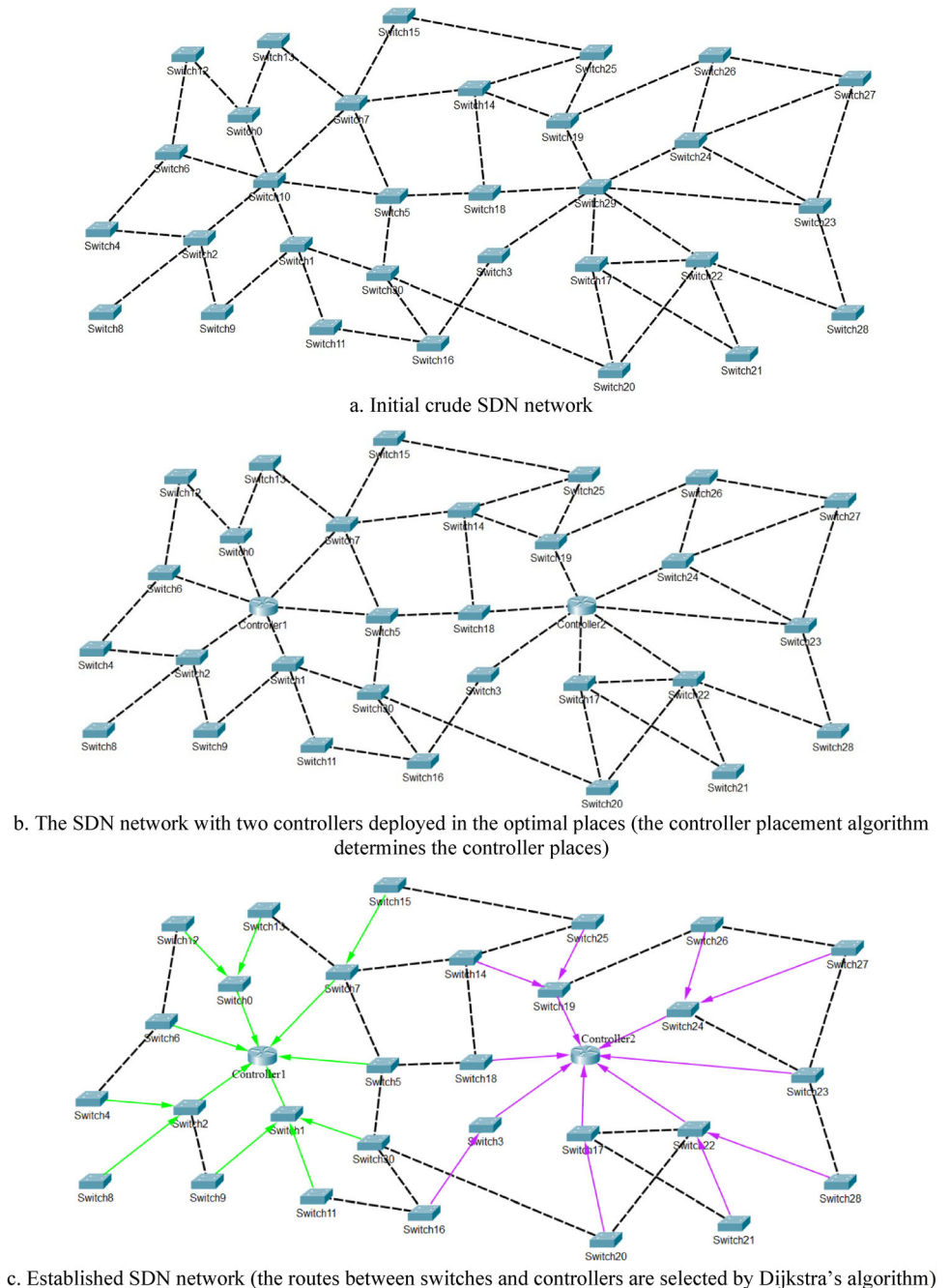
In addition,  $C = \{c_1, c_2, \dots, c_{NC}\}$  is the set of controllers that were installed in the network. The controllers were a sub-set of the switches that were selected as controllers. Firstly, the set of controllers was selected randomly. Afterward, the set is modified by the proposed optimization algorithm to minimize network latency. The

process was encoded, the procedure of which is shown in Fig. 14.

In Fig. 14, the numbers are the IDs of the switches that were selected as controllers. In other words, the 15th, 24th, 3rd, and 10th switches were selected as controllers. The network latency was calculated following the selection of an appropriate sub-set of switches as controllers. The process of calculating the latency is described in subsect. 4.5.

After selecting the controllers, each switch determines the nearest controller using Dijkstra’s shortest path

Fig. 15 The controller placement process



```

Input: Number of Switches (No_Switches), Coordinates of switches (CordinatesDegree), Number of links (No_Links),
details of the links (Links), Adjacency matrix (A), number of solutions (nSol), the maximum number of iterations (Maxiter),
and number of controllers (NC).
Output: A subset of switches as controllers

% Preparing phase
Run Dijkstra's algorithm and Haversine formula to calculate the following requirements:
  Shortest paths between switches (MinPathSwitchToSwitch)
  Minimum distances between switches (MinDistSwitchToSwitch)
  Number of hops between switches (NoHopSwitchToSwitch)

% Initializing phase
for i = 1 to nSol
  Initialize  $X_i$  by equation (23)
  Check  $X_i$  by CheckX function
  Calculate the cost of  $X_i$  by equation (21) ( $F_i$ )
end for

Find BestF and BestX

% Optimizing phase
while iteration < Maxiter
  Divide X into two swarm
  for i = 1 to nSol
    if  $X_i$  is belong to the first swarm
      Update  $X_i$  by proposed discrete MRFO algorithm (newXi)
    else
      Update  $X_i$  by proposed discrete SSA algorithm (newXi)
    end if

    Check newXi by CheckX function
    Calculate cost of newXi by equation (21) (newFi)

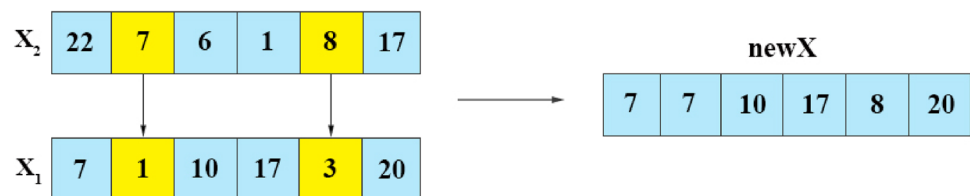
    if newFi <  $F_i$ 
       $F_i = \text{newF}_i$ 
       $X_i = \text{newX}_i$ 
    end if
  end for
  Update BestF and BestX
End while

Return BestX

```

Fig. 16 Pseudocode of the proposed PHCPA controller placement algorithm

Fig. 17 Duplicated indexes in the controller sets



algorithm and sends its data to the nearest controller. It should be noted that there may be more than one path for data transfer between a switch and the nearest controller, which is also selected by Dijkstra's shortest path algorithm. Figure 15 illustrates the proposed controller placement process.

#### 4.5 Objective function

Various competing objective functions were proposed in the literature to evaluate controller placement approaches. In the current paper, the average switch to controller latency was considered as an objective function. The average switch to controller latency is an eminent and widely-used evaluation criterion for controller placement



```

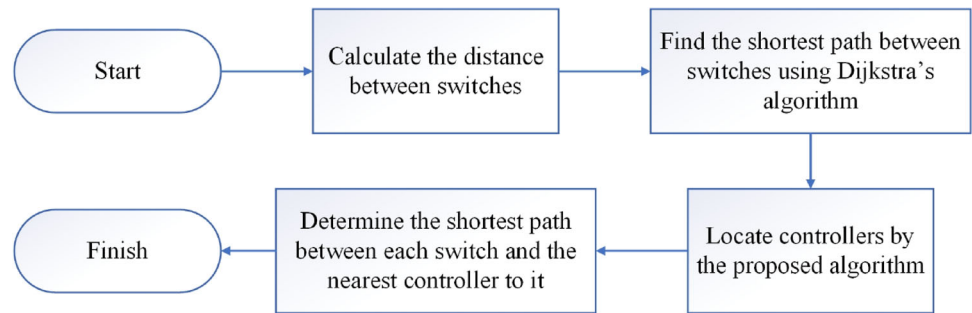
Input:  $X_i, NC$  and set of switches ( $S$ )
Output: corrected  $X_i$ 

 $U = \text{unique}(X_i)$ 
if  $\text{size}(U) < NC$ 
     $D = NC - \text{size}(U)$ 
     $T = S - U$ 
     $W = \text{randsample}(T, D)$ 
     $X_i = U + W$ 
end if

Return  $X_i$ 
    
```

Fig. 18 Pseudocode of CheckX function

Fig. 19 Diagram of the proposed system



used extensively in the literature [22, 68–70]. Equation (21) defines the average switch to controller latency.

$$F(C(\pi)) = \frac{1}{|S|} \sum_{u \in S} (\min_{v \in C} L_{u,v}) \tag{21}$$

where  $|S|$  is the number of switches, and  $L_{u,v}$  is the minimum latency between switch  $u$  and controller  $v$ . The latency could be calculated by Eq. (22) [71].

$$L_{u,v} = \frac{d_{u,v}}{2 * 10^8} \tag{22}$$

where  $d_{u,v}$  is the shortest path length between the switch  $u$  and the controller  $v$ .

#### 4.6 The proposed hybrid controller placement algorithm

This subsection presents the proposed hybrid controller placement algorithm (PHCPA). The hybridization mechanism is a renowned mechanism to increase the optimization algorithms' efficiency and overcome their shortcomings, such as low convergence rate, trapping in the local optimums, poor exploration/exploitation capability, etc. [25, 26, 72]. As previously mentioned, the proposed controller placement algorithm is a network segmentation algorithm, which is considered an optimization problem. The main goal of PHCPA is to minimize the delay in the network by selecting an appropriate subset of switches as controllers. Initially, a random subset of switches was selected as controllers by Eq. (23).

Table 2 Details of the software-defined networks

	Network	Type	Geo extent	Geolocation	Layer	No. switched	No. links
1	ATT North America	COM	Country	USA	IP	25	57
2	BICS	COM	Continent	Europe	IP	33	48
3	Bell Canada	COM	Country+	Canada, USA	IP	48	65
4	Palmetto Net	COM	Region	North Carolina, South Carolina, USA	Fiber	45	70
5	IRIS Networks	COM	Region	Tennessee, USA	IP	51	64
6	SURFNET	REN	Country	Netherlands	IP	50	73

**Table 3** The specifications of the test environment

Name	Value
<b>Hardware</b>	
CPU	Core i7
Frequency	2.7 GHz
RAM	8 GB
Hard drive	750 GB
<b>Software</b>	
Operating system	Windows 10
Language	MATLAB R2017a

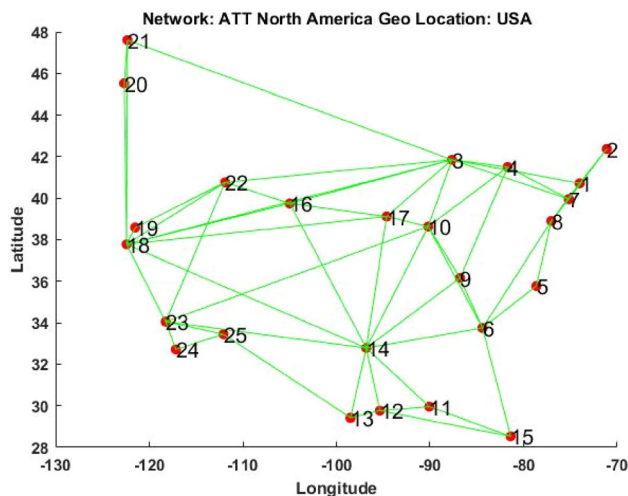
**Table 4** The parameter values of the algorithms

Algorithm	Parameter	value
Proposed solution	<i>Alpha</i>	50
	<i>Beta</i>	0.2
	$\alpha_1, \alpha_2, \alpha_3$ and $\alpha_4$	0.25,0.5,0.75,1
	<i>Threshold</i>	10
MRFO	$\alpha$	$2r \times \sqrt{ \log(r) }$
	$r_1, r_2, r_3,$ and $r$	rand
	$\omega$	rand
	$s$	2
	MFO	$a$
MVO	$b$	- 1 to - 2
	<i>WEPMin</i>	0.2
SCA	<i>WEPMax</i>	1
	$p$	6
	$r_1, r_2, r_3$ and $r_4$	rand
SSA	$c_1, c_2, c_3$	rand
WOA	$r_1, r_2,$ and $p$	rand
	$a$	2 to 0
	$C$	$2 \times r_2$

\*Rand function generates a random number in [0,1]

$$X_i = \text{floor}(\vec{r}\vec{v} * NS) + 1 \tag{23}$$

where  $\vec{r}\vec{v}$  is a vector with  $NC$  random numbers in  $[0, 1]$ ,  $NC$  is the number of controllers,  $NS$  is the number of network controllers, and  $\text{floor}$  is a function that rounds the decimal number to the nearest smaller integer. Following the generation of random subsets of controllers, the subsets were modified by the proposed hybrid discrete MRFO-SSA algorithm to find the best subset of controllers to minimize the network delay. In each round of the proposed PHCPA algorithm, the solutions were divided into two swarms; discrete MRFO modified the first swarm controller sets. Discrete SSA modified the solutions in the second swarm.



**Fig. 20** ATT North America network

Figure 16 presents the pseudocode of PHCPA. In modifying controller sets, it is possible to place duplicate indexes in a set. For instance, Fig. 17 represents this situation in using a two-point swap operator.

It is evident from Fig. 17 that in the resulting set, the index seven was duplicated, and the number of controllers was reduced from 6 to 5. A mechanism was needed to solve this unacceptable situation. Therefore, the *CheckX* function was proposed in the PHCPA algorithm to check the controllers’ set, the pseudocode of which is illustrated in Fig. 18. Furthermore, the diagram of the proposed system is provided in Fig. 19.

### 4.7 Computational complexity

This subsection discusses the computational complexity of the proposed algorithm. The proposed algorithm consists of three main parts: calculating the distances between switches, finding the shortest path between switches, finding the optimal controller places.

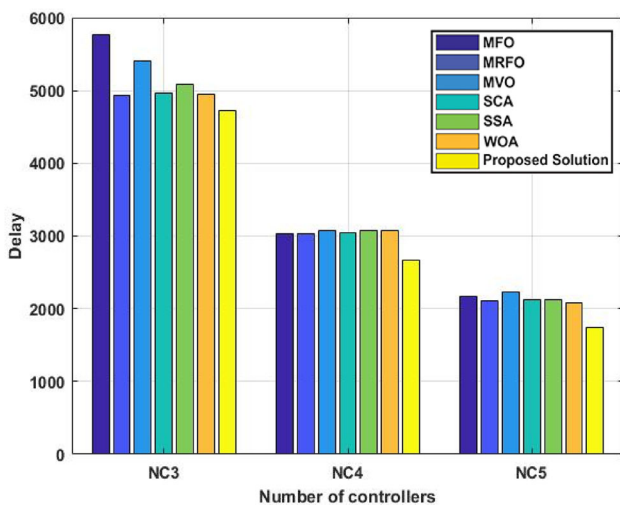
The computational complexity of the distance calculation step is  $O(S^2)$ , where  $S$  is the number of switches. The complexity of the basic Dijkstra’s algorithm, which finds the shortest path between two points, is  $O(S + E \times \log(S))$ . Therefore, the computational complexity of calculating the shortest path between all switches is  $O(S \times (S - 1) \times (S + E) \times \log(S))$ , where the  $E$  is the number of the network’s communication links.

The computational complexity of finding optimal controller places could be argued in three partial parts (i.e., initialization, discrete SSA, and discrete MRFO). The complexity of the initialization is  $O(N^2)$ , where  $N$  is the number of solutions. The computational complexity of the proposed discrete SSA algorithm is  $O(R \times N \times NC)$ , where  $R$  is the number of iterations, and  $NC$  is the number

**Table 5** Statistical results of the algorithms obtained on the ATT North America network over 30 independent runs in terms of network delay

	Minimum delay	Maximum delay	Average delay	Standard deviation
NC = 3				
MFO	4.89907E+03	5.76801E+03	5.00768E+03	2.43932E+02
MRFO	4.89907E+03	4.93297E+03	4.90230E+03	1.01982E+01
MVO	4.89907E+03	5.40630E+03	4.91936E+03	1.01447E+02
SCA	4.89907E+03	4.97006E+03	4.91247E+03	2.01728E+01
SSA	4.89907E+03	5.08041E+03	4.92379E+03	4.49382E+01
WOA	4.89907E+03	4.94743E+03	4.91182E+03	1.89237E+01
<b>Proposed solution</b>	<b>4.71658E+03</b>	<b>4.71658E+03</b>	<b>4.71658E+03</b>	<b>1.86624E-12</b>
NC = 4				
MFO	2.69808E+03	3.02845E+03	2.94601E+03	9.79129E+01
MRFO	2.80408E+03	3.02845E+03	2.94957E+03	7.71242E+01
MVO	2.69808E+03	3.07096E+03	2.99703E+03	8.64070E+01
SCA	2.77865E+03	3.04947E+03	2.98215E+03	8.08606E+01
SSA	2.83129E+03	3.07096E+03	3.01078E+03	4.83404E+01
WOA	2.72351E+03	3.07096E+03	2.94257E+03	1.08698E+02
<b>Proposed solution</b>	<b>2.66546E+03</b>	<b>2.66546E+03</b>	<b>2.66546E+03</b>	<b>0.00000E+00</b>
NC = 5				
MFO	1.99229E+03	2.16847E+03	2.08075E+03	4.73380E+01
MRFO	1.79142E+03	2.10336E+03	2.00583E+03	7.62164E+01
MVO	1.99229E+03	2.22521E+03	2.06352E+03	6.31692E+01
SCA	1.97195E+03	2.12353E+03	2.04684E+03	4.03719E+01
SSA	1.97195E+03	2.12685E+03	2.05341E+03	3.91505E+01
WOA	1.97195E+03	2.07657E+03	2.03394E+03	3.44417E+01
<b>Proposed solution</b>	<b>1.74550E+03</b>	<b>1.74550E+03</b>	<b>1.74550E+03</b>	<b>6.99841E-13</b>

The best results are written in bold



**Fig. 21** Bar graphs of the maximum delay obtained by the algorithms on the ATT North America network

of controllers. The complexity of the proposed discrete MRFO algorithm is also  $O(R \times N \times NC)$ .

It should be noted that the proposed controller placement algorithm is a multi-swarm algorithm, the solutions of which were divided into two swarms in each iteration, and

each swarm was updated with one of the discrete algorithms. Therefore, the overall computational complexity of the optimal location discovery of the controller selection part could not be calculated accurately. However, since the computational complexity of discrete SSA and MRFO algorithms was almost similar, the complexity of this part was approximately equal to  $O(R \times N^3 \times NC)$ .

### 5 Experimental result

This section presents a bulk of experiments' results to evaluate the proposed algorithm's performance and substantiate the previous section's contributions. The experiments were conducted on six real-world networks (see Table 2), and the results of the proposed algorithm was compared with the results of several newfangled algorithms containing the Moth-Flame Optimization algorithm (MFO) [73], Manta-Ray Foraging Optimization algorithm (MRFO) [57], Multi-Verse Optimizer (MVO) [74], Sine Cosine Algorithm (SCA) [75], Salp Swarm Algorithm (SSA) [59], and Whale Optimization Algorithm (WOA) [58]. Furthermore, due to the random nature of the

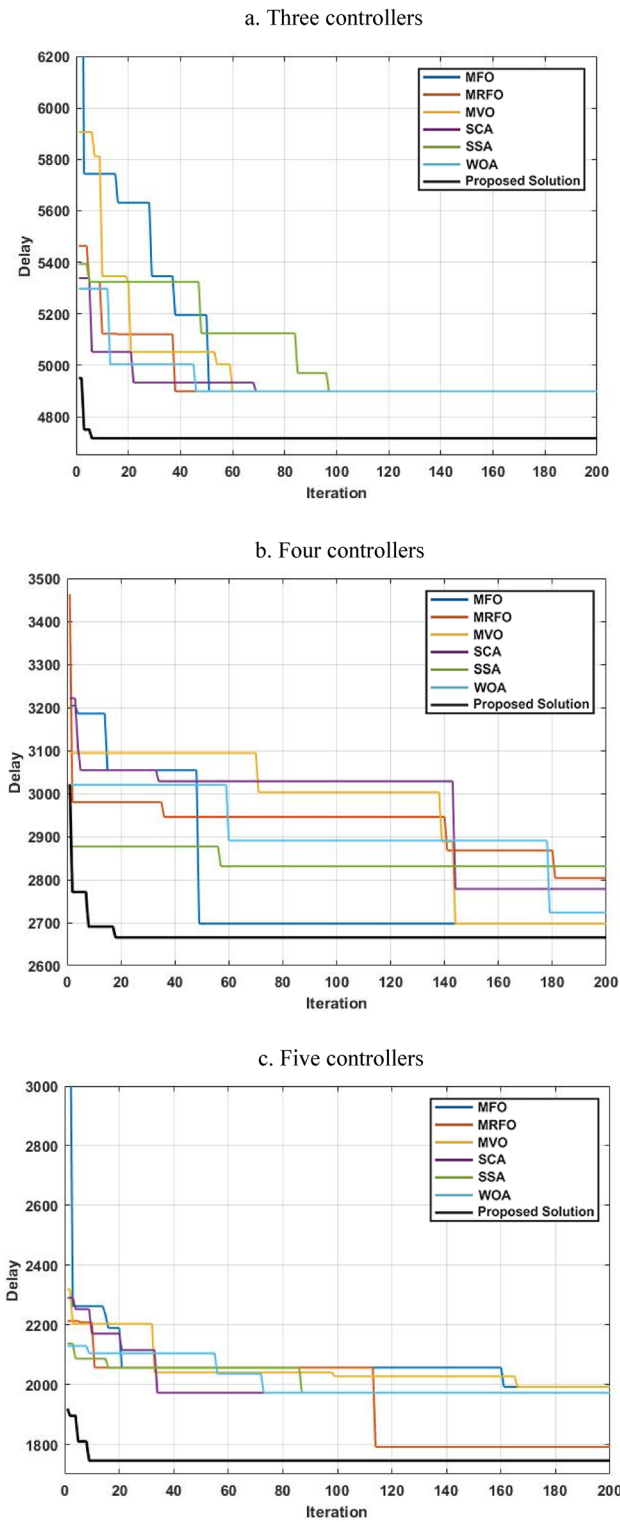


Fig. 22 Convergence rates of the algorithms on the ATT North America network

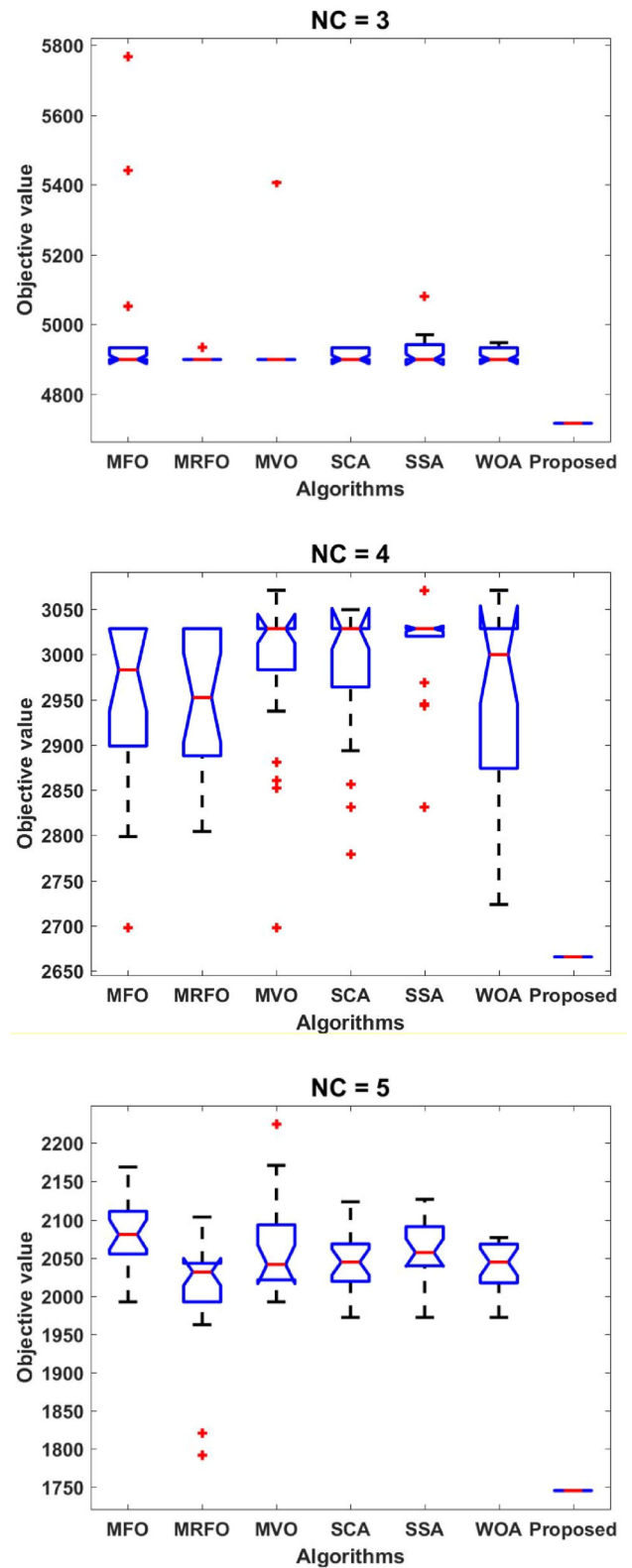


Fig. 23 Box plots of the algorithms on the ATT North America network

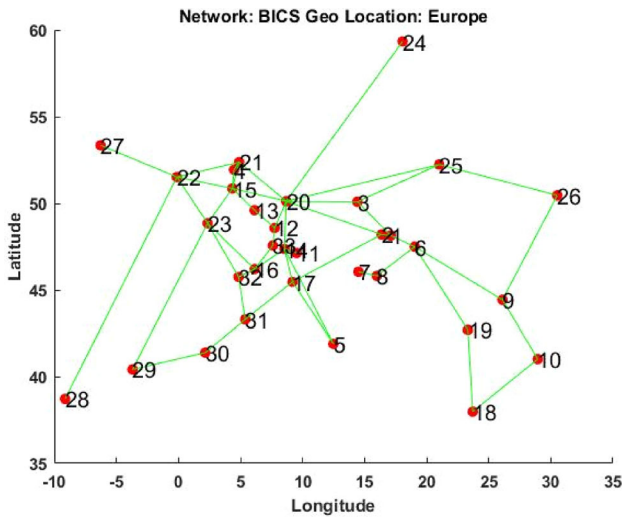


Fig. 24 The BICS network

optimization algorithms mentioned above, each algorithm was run 30 times independently, and the results were presented statistically/visually.

It is worth mentioning that all experiments were carried out in the same environment with specifications expressed in Table 3. The number of solutions and the maximum

number of iterations in the algorithms were set to 50 and 200, respectively. The other parameter values of the algorithms are shown in Table 4, which are the algorithms’ default values. Moreover, reaching the maximum number of iterations was considered as a stopping criterion.

### 5.1 Experiments on ATT North America network

This subsection presents the obtained results by the algorithms on the ATT North America network. As provided in Table 2, this network possesses twenty-two switches and fifty-seven links between them. Figure 20 illustrates the network and the position of the switches.

The statistical results of the algorithms obtained on this network for a different number of controllers are presented in Table 5. The bar graphs of the maximum delay obtained by the algorithms are represented in Fig. 21.

It can be deduced from Table 5 and Fig. 21 that the proposed algorithm achieved better results than other competing algorithms in terms of minimum, maximum, average delay, and standard deviation of obtained results on three different numbers of controllers in the network. For further investigation of the algorithm’s performance, the algorithms’ best convergence rates for different

**Table 6** Statistical results of the algorithms obtained on the BICS network over 30 independent runs in terms of network delay

	Minimum delay	Maximum delay	Average delay	Standard deviation
NC = 3				
MFO	7.62891E+03	9.18170E+03	7.74809E+03	3.57502E+02
MRFO	7.62891E+03	7.76427E+03	7.66786E+03	6.00839E+01
MVO	7.62891E+03	7.90206E+03	7.65630E+03	7.11758E+01
SCA	7.62891E+03	7.76427E+03	7.68111E+03	6.58999E+01
SSA	7.62891E+03	7.90206E+03	7.67229E+03	8.13971E+01
WOA	7.62891E+03	7.77210E+03	7.65847E+03	5.64859E+01
<b>Proposed solution</b>	<b>7.32801E+03</b>	<b>7.32801E+03</b>	<b>7.32801E+03</b>	<b>0.00000E+00</b>
NC = 4				
MFO	5.00117E+03	5.33001E+03	5.02991E+03	8.00776E+01
MRFO	4.94928E+03	5.01219E+03	4.99782E+03	1.39307E+01
MVO	5.00117E+03	5.12409E+03	5.02935E+03	5.11534E+01
SCA	4.84776E+03	5.12409E+03	4.99923E+03	4.57347E+01
SSA	4.94133E+03	5.12409E+03	5.02739E+03	5.90763E+01
WOA	4.95011E+03	5.12409E+03	5.00239E+03	3.26732E+01
<b>Proposed solution</b>	<b>4.84776E+03</b>	<b>4.84776E+03</b>	<b>4.84776E+03</b>	<b>1.00066E-12</b>
NC = 5				
MFO	3.59052E+03	3.73254E+03	3.66766E+03	4.29993E+01
MRFO	3.59052E+03	3.79270E+03	3.71041E+03	5.63543E+01
MVO	3.59052E+03	3.81460E+03	3.67786E+03	7.17170E+01
SCA	3.59968E+03	3.79828E+03	3.69391E+03	6.17855E+01
SSA	3.54151E+03	3.82040E+03	3.66487E+03	6.77810E+01
WOA	3.54151E+03	3.75345E+03	3.64412E+03	6.42126E+01
<b>Proposed solution</b>	<b>3.43070E+03</b>	<b>3.43070E+03</b>	<b>3.43070E+03</b>	<b>1.39968E-12</b>



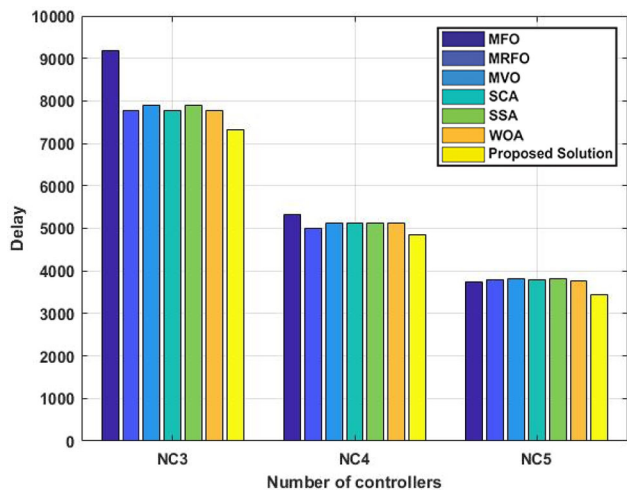


Fig. 25 Bar graphs of the maximum delay obtained by the algorithms on the BICS network

numbers of controllers were compared, and the results are illustrated in Fig. 22.

According to the graphs of Fig. 22, it can be observed that the proposed solution converged earlier and found a better solution than other algorithms. Moreover, Fig. 23 represents the box plots of the algorithms on the ATT North America Network. It can be concluded from the box plots of Fig. 23, the proposed algorithm outperformed the competitor algorithms and had a lesser random state.

### 5.2 Experiments on BICS network

The BICS software-defined network has thirty-three switches and forty-eight links, which are shown in Fig. 24. The details of the BICS network are stated in Table 2. This subsection presents the results of experiments on this network.

The statistical results of the algorithms obtained from this network for the different number of controllers are presented in Table 6. Besides, the bar graphs of the maximum delay obtained by the algorithms are represented in Fig. 25.

According to the results of Table 6 and Fig. 25, the proposed algorithm outperformed competitor algorithms and decreased the minimum, maximum, and average delays of the network on three different numbers of controllers. Additionally, Fig. 26 shows the convergence rates of the algorithms for the different numbers of controllers.

The simulation results expressed that the proposed solution achieved better results in a shorter time than competitor algorithms. Additionally, the algorithms’ box plots on this network are provided in Fig. 27 for more-in-depth comparisons. These box graphs showed that competing algorithms’ results were very scattered, especially at

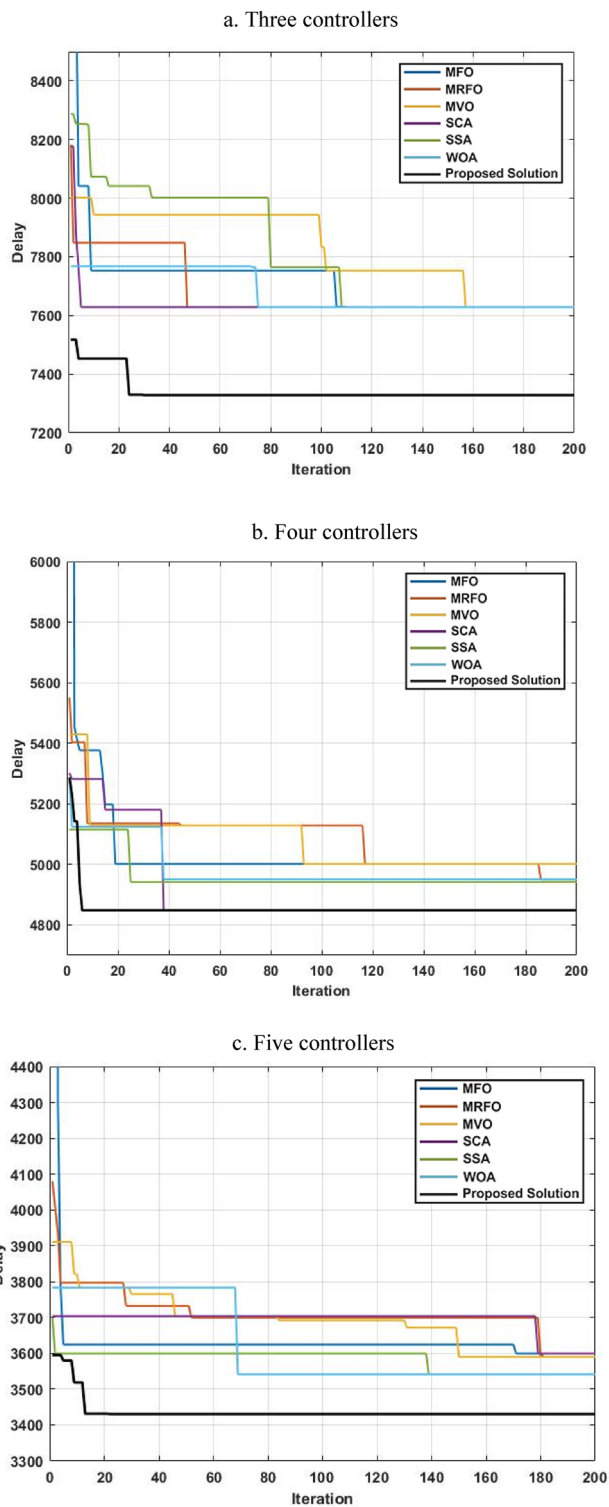


Fig. 26 Convergence rates of the algorithms on the BICS network

NC = 5, which was more complicated. Notwithstanding, the proposed algorithm results in all three scenarios were more consistent, which could be stated that the proposed algorithm was better than competing algorithms.

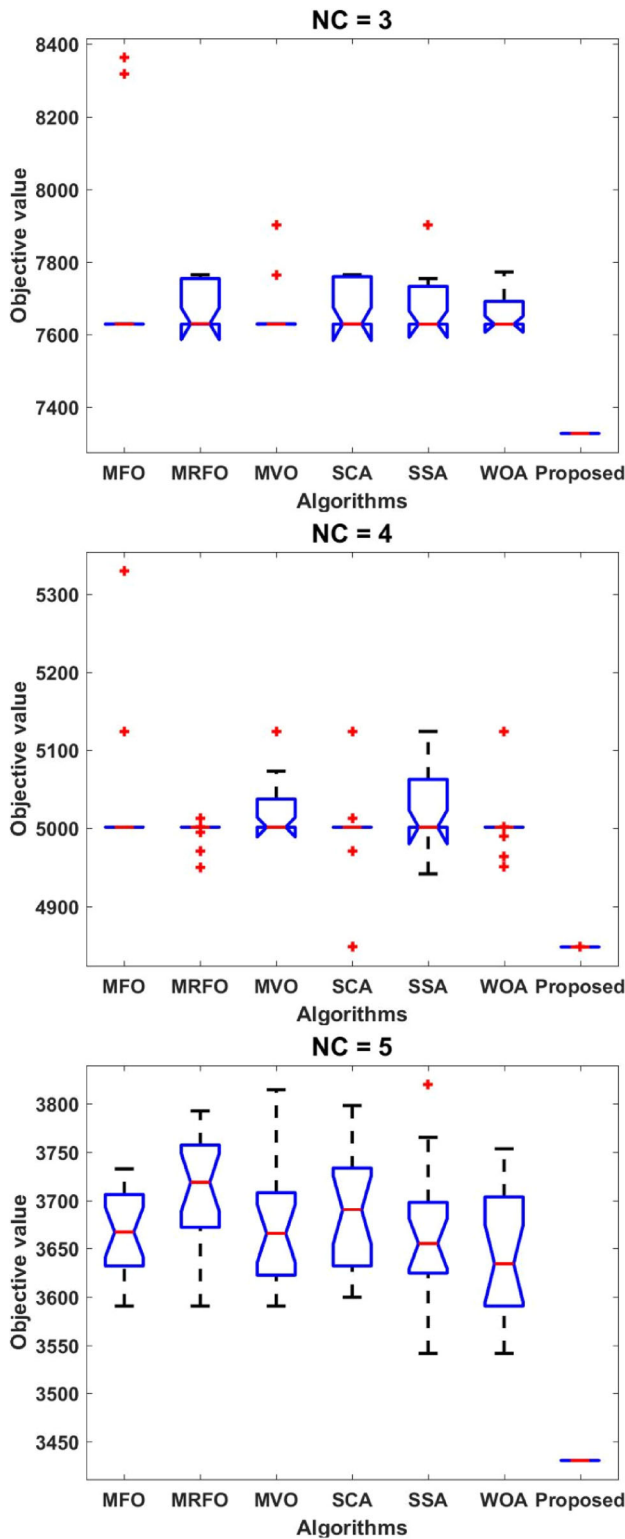


Fig. 27 Box plots of the algorithms on the BICS network

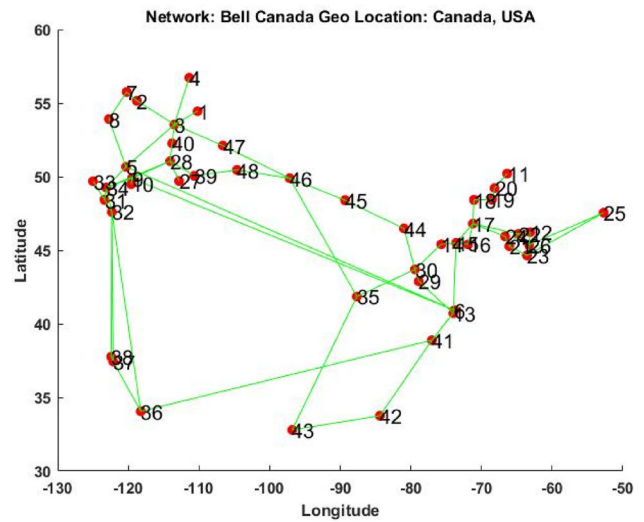


Fig. 28 The Bell Canada network

### 5.3 Experiments on Bell Canada network

This subsection evaluates the algorithms’ performance on the Bell Canada network and provides the statistical and visual results. The Bell Canada network consists of forty-eight switches in Canada and the USA and sixty-five communication links. The geolocation of the switches and the details of the links are exhibited in Fig. 28.

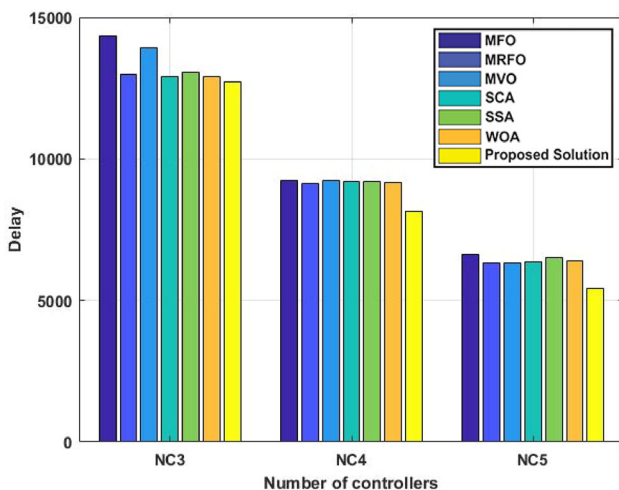
The statistical results of the algorithms obtained on this network for a different number of controllers are presented in Table 7. Moreover, the bar graphs of the maximum delay obtained by the algorithms are represented in Fig. 29.

The statistical results of Table 7 demonstrated that all algorithms found the optimal places of controllers of the network with three controllers. Nonetheless, by increasing the number of network controllers, the algorithms lost their efficiency and could not find the optimal places. This phenomenon can be due to the fact that the increase in the number of controllers surges the complexity of finding optimal places. However, by examining the results of Table 7 and the graphical results of Fig. 29, it can be concluded that the proposed algorithm performed better than competitor algorithms and reduced the delay of the network by finding optimal places for deploying the controllers. Like previous subsections, the algorithms’ convergence rate was investigated, and the results are provided in Fig. 30 for the different number of controllers.

It could be inferred from the algorithms’ results that the proposed solution outperformed the competitor algorithms and reduced network delay on the Bell Canada network. Furthermore, Fig. 31 illustrates the box plots of the

**Table 7** Statistical results of the algorithms obtained on the Bell Canada network over 30 independent runs in terms of network delay

	Minimum delay	Maximum delay	Average delay	Standard deviation
NC = 3				
MFO	<b>1.27271E+04</b>	1.43490E+04	1.28374E+04	3.57320E+02
MRFO	<b>1.27271E+04</b>	1.29823E+04	1.28019E+04	7.29135E+01
MVO	<b>1.27271E+04</b>	1.39273E+04	1.28776E+04	3.54145E+02
SCA	<b>1.27271E+04</b>	1.29161E+04	1.27825E+04	5.80453E+01
SSA	<b>1.27271E+04</b>	1.30494E+04	1.27983E+04	9.12127E+01
WOA	<b>1.27271E+04</b>	1.29019E+04	1.27813E+04	5.61810E+01
<b>Proposed solution</b>	<b>1.27271E+04</b>	<b>1.27271E+04</b>	<b>1.27271E+04</b>	<b>3.73249E-12</b>
NC = 4				
MFO	8.20088E+03	9.23515E+03	8.78858E+03	3.94639E+02
MRFO	8.20088E+03	9.12972E+03	8.66577E+03	2.84317E+02
MVO	8.18164E+03	9.22955E+03	8.81791E+03	3.49735E+02
SCA	8.22794E+03	9.19873E+03	8.74866E+03	3.10148E+02
SSA	8.16546E+03	9.19873E+03	8.56902E+03	3.40545E+02
WOA	8.11478E+03	9.15080E+03	8.56192E+03	2.70457E+02
<b>Proposed solution</b>	<b>8.07990E+03</b>	<b>8.15020E+03</b>	<b>8.08693E+03</b>	<b>2.16371E+01</b>
NC = 5				
MFO	6.03077E+03	6.61245E+03	6.18162E+03	1.48477E+02
MRFO	6.00638E+03	6.31909E+03	6.18143E+03	1.09967E+02
MVO	6.01923E+03	6.32641E+03	6.14745E+03	8.83645E+01
SCA	6.09740E+03	6.35779E+03	6.19970E+03	9.26544E+01
SSA	6.01103E+03	6.53077E+03	6.14597E+03	1.19036E+02
WOA	6.01103E+03	6.38567E+03	6.17269E+03	9.96717E+01
<b>Proposed solution</b>	<b>5.37059E+03</b>	<b>5.42683E+03</b>	<b>5.37394E+03</b>	<b>1.26787E+01</b>



**Fig. 29** Bar graphs of the maximum delay obtained by the algorithms on the Bell Canada network

algorithms on this network. In the guise of the box graphs of Fig. 31, the proposed method's network latency was less than other algorithms. The difference between the algorithms increased with the growing complexity (raising the number of controllers).

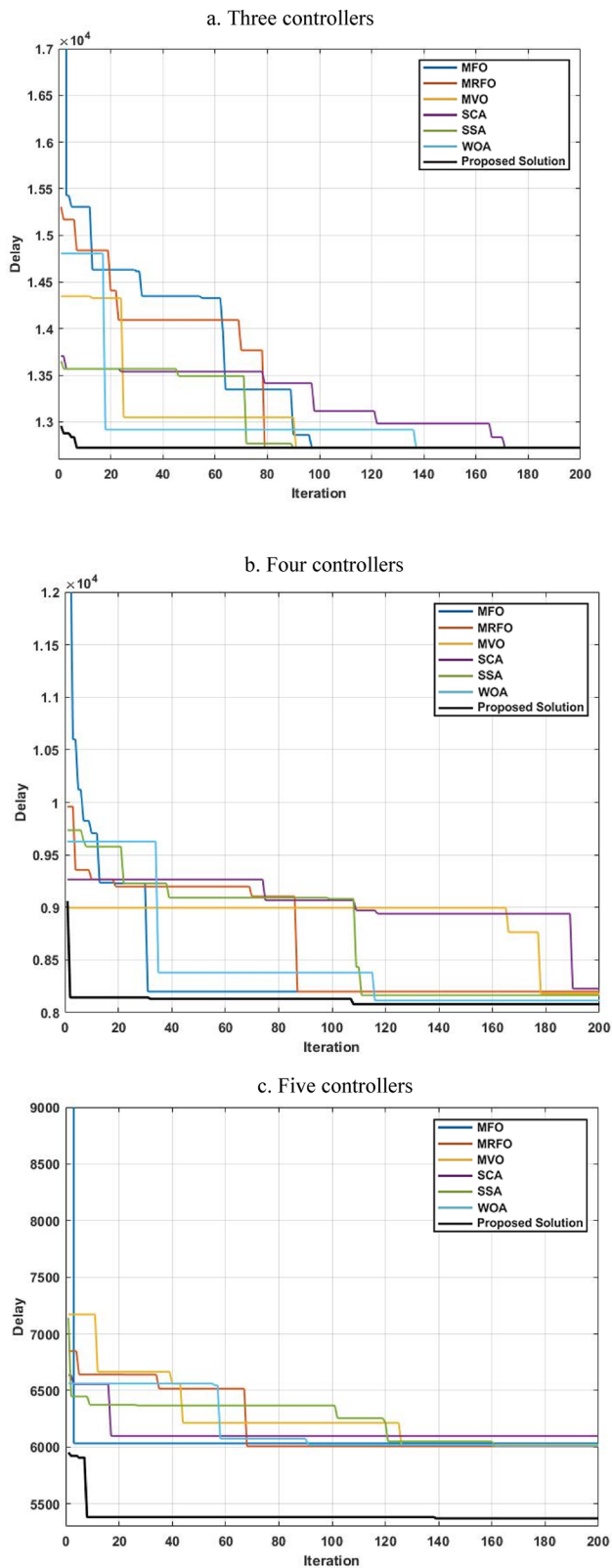
## 5.4 Experiments on Palmetto Net network

Herein, the experimental results of the algorithms on the Palmetto Net network are presented. The Palmetto Net network is a regional network and has forty-five switches with seventy links. Figure 32 represents the Palmetto Net network.

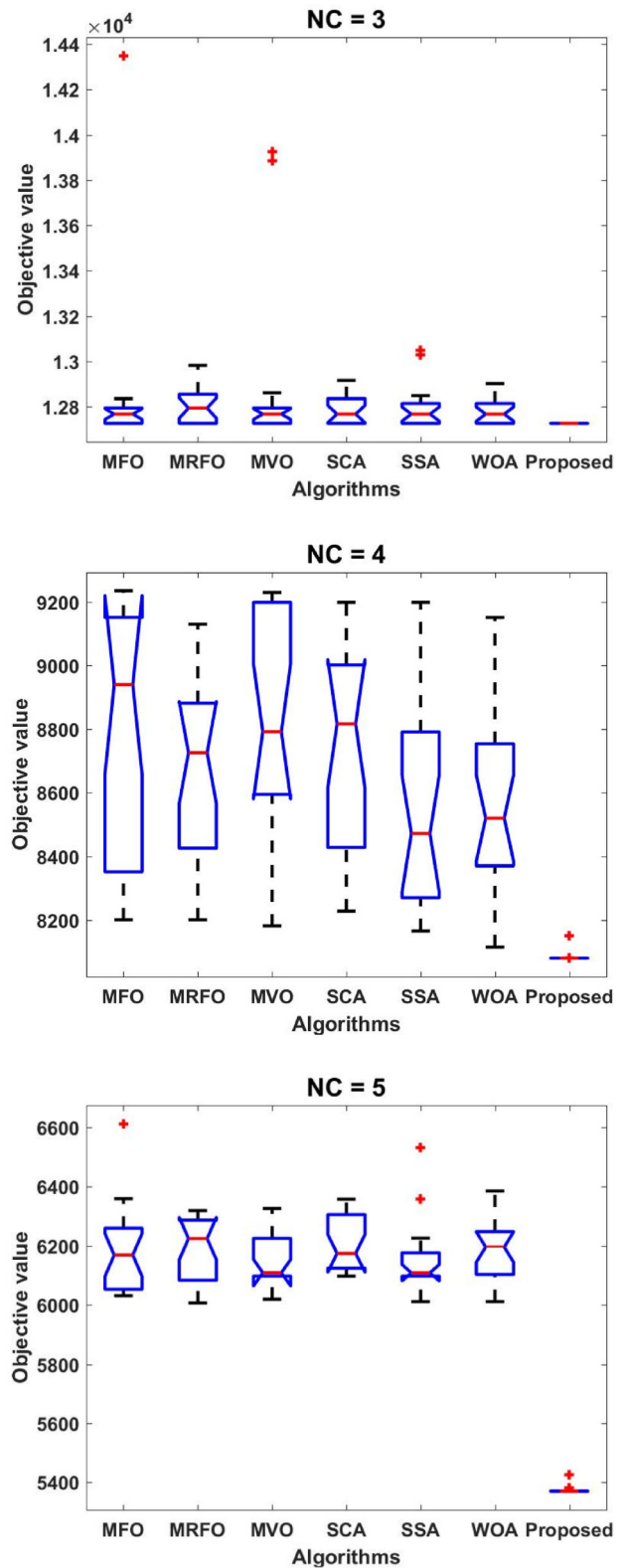
Table 8 presents the statistical results of the algorithms on the Palmetto Net network. The results are provided for three different numbers of controllers: 3, 4, and 5. Besides, Fig. 33 illustrates the bar graphs of the worst results of the algorithms on this network.

Table 8 and Fig. 33 express that the proposed solution algorithm found better places to establish controllers and significantly reduced network delay. For having a more in-depth evaluation algorithm, the converge rate analysis was done for the algorithms, and the results are provided in Fig. 34. The graphs of Fig. 34 corroborated Table 8 and imparted the superiority of the proposed algorithm.

Additionally, for a more detailed examination of the algorithms, the algorithms' box plots on the Palmetto Net network are presented in Fig. 35. Box plots of Fig. 35 confirmed the statistical results of Table 8, bar diagrams of



**Fig. 30** Convergence rates of the algorithms on the Bell Canada network



**Fig. 31** Box plots of the algorithms on the Bell Canada network

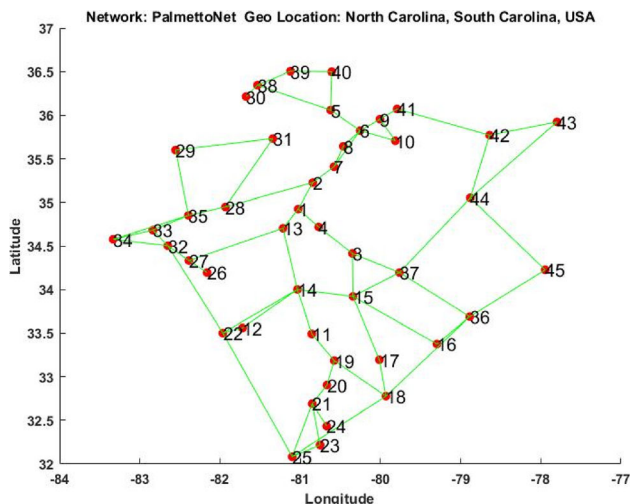


Fig. 32 The Palmetto Net network

Fig. 33, and convergence diagrams of Fig. 34 and revealed the proposed algorithm’s superiority.

### 5.5 Experiments on IRIS network

The IRIS network is another regional software-defined network, which was speared in Tennessee and the USA. The IRIS network has fifty-one switches and sixty-four communicational links among them (see Table 2). Figure 36 plots the IRIS network and the details of switches and links of it.

The calculated minimum, maximum, and average delays obtained by the algorithms on the IRIS network are stated in Table 9. Furthermore, Fig. 37 demonstrated the bar graphs of the maximum network delay obtained by the algorithms.

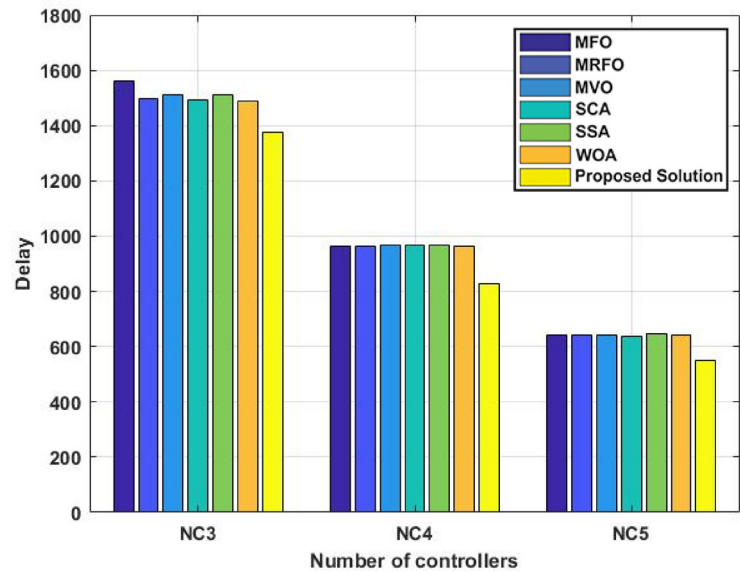
As shown in Table 9 and Fig. 37, the proposed algorithm achieved better results than other competing algorithms in terms of minimum, maximum, average delay, and standard deviation of obtained results on three different numbers of controllers in the network. Besides, for a more in-depth evaluation of the algorithms’ performance, the algorithms’ best convergence rates for different numbers of

Table 8 Statistical results of the algorithms obtained on the Palmetto Net network over 30 independent runs in terms of network delay

	Minimum delay	Maximum delay	Average delay	Standard deviation
NC = 3				
MFO	1.44611E+03	1.56112E+03	1.47934E+03	2.72078E+01
MRFO	1.44611E+03	1.49695E+03	1.46465E+03	1.96586E+01
MVO	1.44611E+03	1.51418E+03	1.47676E+03	2.13208E+01
SCA	1.44611E+03	1.49331E+03	1.46896E+03	1.83495E+01
SSA	1.44611E+03	1.51418E+03	1.46818E+03	2.45536E+01
WOA	1.44611E+03	1.48945E+03	1.45661E+03	1.68394E+01
<b>Proposed solution</b>	<b>1.37381E+03</b>	<b>1.37381E+03</b>	<b>1.37381E+03</b>	<b>0.00000E+00</b>
NC = 4				
MFO	8.91431E+02	9.64610E+02	9.42772E+02	2.19623E+01
MRFO	8.87096E+02	9.64091E+02	9.33757E+02	2.33985E+01
MVO	8.86522E+02	9.69537E+02	9.35537E+02	2.64041E+01
SCA	8.84674E+02	9.68244E+02	9.21847E+02	2.23344E+01
SSA	8.81320E+02	9.69537E+02	9.28403E+02	2.80014E+01
WOA	8.68546E+02	9.64610E+02	9.26136E+02	3.45958E+01
<b>Proposed solution</b>	<b>8.29219E+02</b>	<b>8.29219E+02</b>	<b>8.29219E+02</b>	<b>3.49921E-13</b>
NC = 5				
MFO	5.69360E+02	6.42896E+02	6.11257E+02	2.08642E+01
MRFO	5.75789E+02	6.43437E+02	6.12968E+02	2.31472E+01
MVO	5.71628E+02	6.43660E+02	6.19196E+02	2.67308E+01
SCA	5.88314E+02	6.38955E+02	6.14010E+02	1.87298E+01
SSA	5.71710E+02	6.47601E+02	6.07425E+02	2.31739E+01
WOA	5.71169E+02	6.41659E+02	6.07435E+02	2.22237E+01
<b>Proposed solution</b>	<b>5.52983E+02</b>	<b>5.52983E+02</b>	<b>5.52983E+02</b>	<b>4.51746E-14</b>



**Fig. 33** Bar graphs of the maximum delay obtained by the algorithms on the Palmetto Net network



controllers were compared, the results of which are illustrated in Fig. 38.

The convergence rate graphs of Fig. 38 represent that the proposed solution discovered better solutions in a shorter time compared to other state-of-the-art algorithms.

Figure 39 presents the box plots of the algorithms on the IRIS network. Regarding these box diagrams, it could be easily comprehended that competitor algorithms achieved a different network latency level in the independent runs, unlike the proposed algorithm, which refutes these algorithms' effectiveness. However, by examining the box diagrams, it could be claimed that the proposed algorithm was reliable and could reduce network latency.

## 5.6 Experiments on SURFNET network

In this subsection, the algorithms were evaluated on a giant software-defined network to examine the algorithms' scalability. Finally, the algorithms were investigated in terms of their performance as well. The last network was the SURFNET network, which had fifty switches and seventy-three communicational links. The details of this network are presented in Table 10 and Fig. 40.

The minimum, maximum, and average network delay of the algorithms on the SURFNET network for three different numbers of controllers are represented in Table 10. Moreover, Table 10 provides the standard deviations of the results of the algorithms. Additionally, Fig. 41 depicts the maximum delays of the network obtained by the algorithms using bar graphs.

According to Table 10 and Fig. 41, it could be seen that the proposed algorithm demonstrated a significant difference from the competitor algorithms and achieved better results. The best convergence rates of the algorithms on the different number of controllers are presented in Fig. 42.

Likewise, the box plot graphs of the algorithms for the SURFNET network are exhibited in Fig. 43. Concerning the box graphs of Fig. 43, it could be perceived that the proposed algorithm had a significant difference from the competitor algorithms and depreciated the network latency as much as possible. Furthermore, it was observed that the results of the proposed algorithm were more coherent compared to the results of other algorithms, which is an indicator of the proposed algorithm's superiority.

The algorithms' experimental results on various software-defined networks revealed the proposed controller

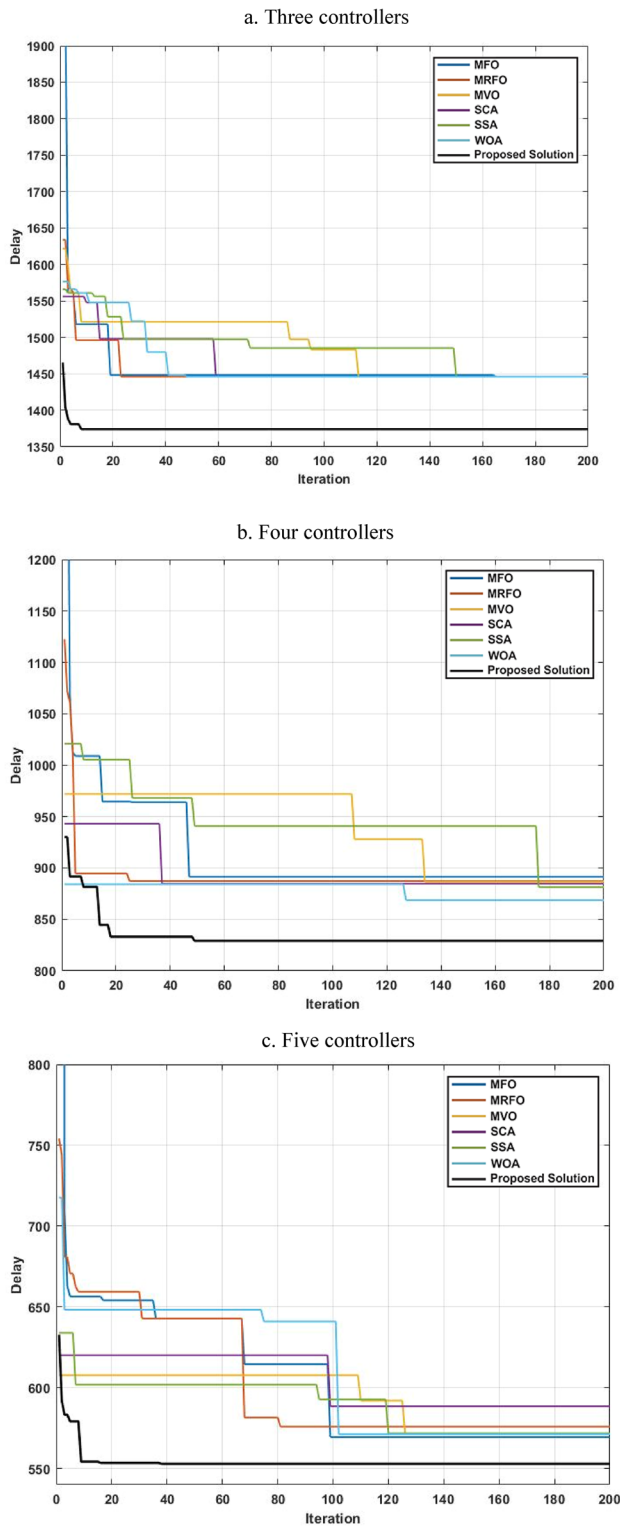


Fig. 34 Convergence rates of the algorithms on the Palmetto Net network

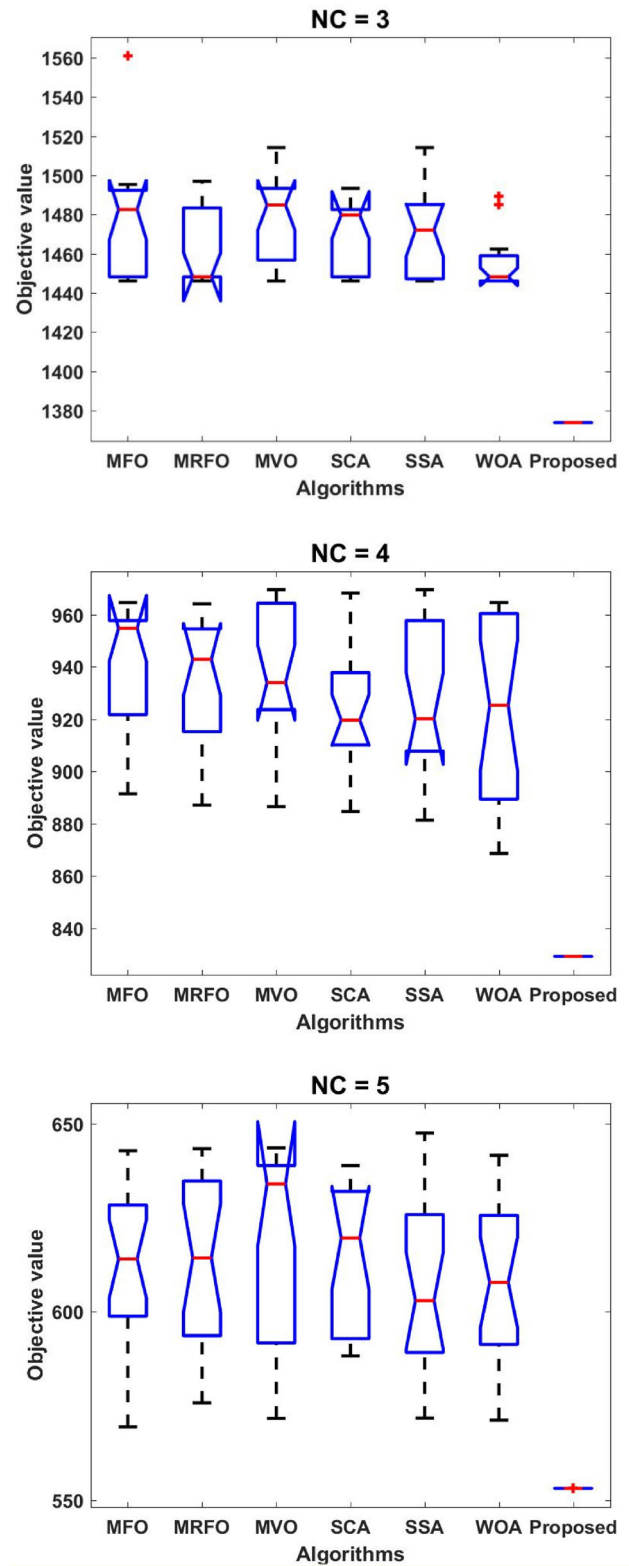


Fig. 35 Box plots of the algorithms on the Palmetto Net network

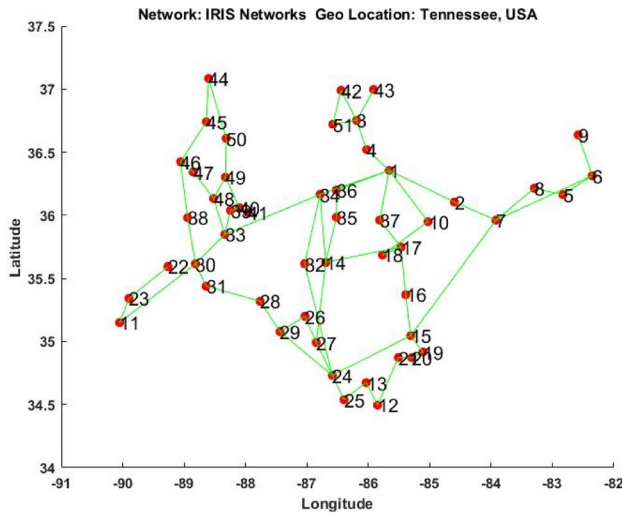


Fig. 36 The IRIS network

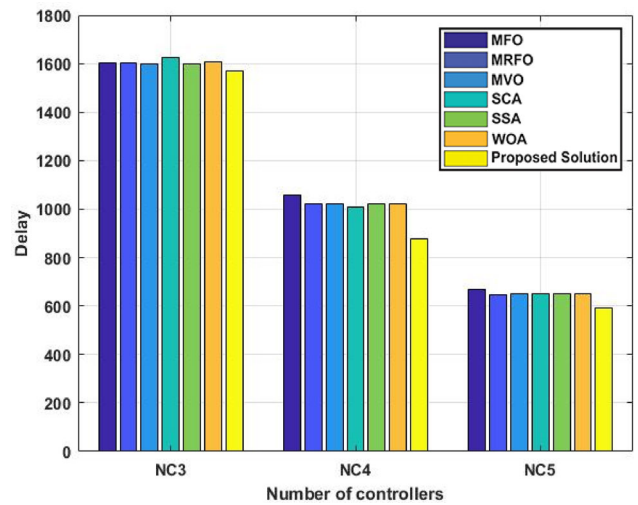
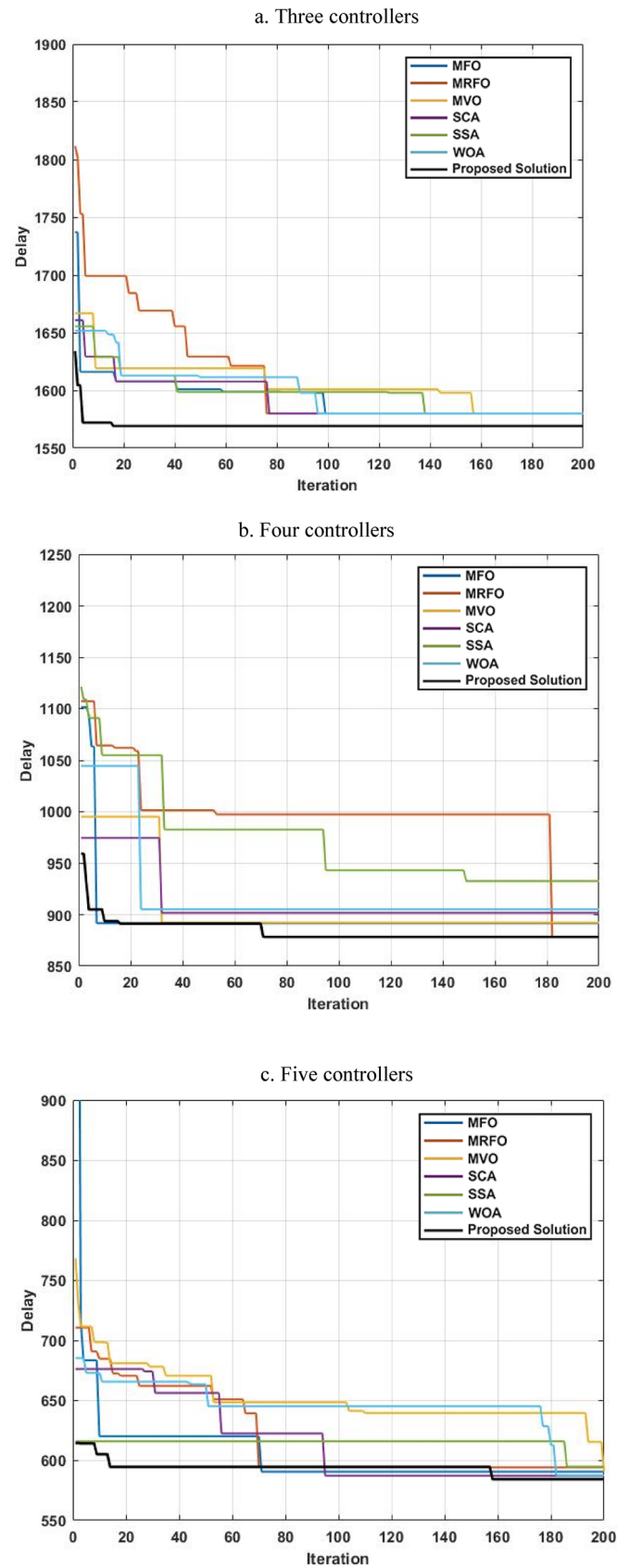


Fig. 37 Bar graphs of the maximum delay obtained by the algorithms on the IRIS network

Table 9 Statistical results of the algorithms obtained on the IRIS network over 30 independent runs in terms of network delay

	Minimum delay	Maximum delay	Average delay	Standard deviation
NC = 3				
MFO	1.58010E+03	1.60090E+03	1.58394E+03	7.40214E+00
MRFO	1.58010E+03	1.60090E+03	1.58472E+03	8.09729E+00
MVO	1.58010E+03	1.59779E+03	1.58419E+03	6.87558E+00
SCA	1.58010E+03	1.62511E+03	1.59133E+03	1.33278E+01
SSA	1.58010E+03	1.59853E+03	1.58643E+03	8.15688E+00
WOA	1.58010E+03	1.60505E+03	1.58344E+03	7.31950E+00
<b>Proposed solution</b>	<b>1.56925E+03</b>	<b>1.56925E+03</b>	<b>1.56925E+03</b>	<b>4.66561E-13</b>
NC = 4				
MFO	8.91702E+02	1.05764E+03	9.83354E+02	4.01508E+01
MRFO	<b>8.78431E+02</b>	1.02059E+03	9.61357E+02	3.93573E+01
MVO	8.92278E+02	1.02059E+03	9.76144E+02	4.32017E+01
SCA	9.01764E+02	1.00806E+03	9.49383E+02	3.63860E+01
SSA	9.32590E+02	1.02059E+03	9.74307E+02	2.59789E+01
WOA	9.05409E+02	1.02178E+03	9.77150E+02	3.27428E+01
<b>Proposed solution</b>	<b>8.78431E+02</b>	<b>8.78431E+02</b>	<b>8.78431E+02</b>	<b>9.40384E-14</b>
NC = 5				
MFO	5.90395E+02	6.69925E+02	6.19809E+02	2.17945E+01
MRFO	5.93953E+02	6.45621E+02	6.20318E+02	1.26785E+01
MVO	5.90580E+02	6.52745E+02	6.26911E+02	1.84837E+01
SCA	5.87022E+02	6.52745E+02	6.21648E+02	1.65607E+01
SSA	5.94564E+02	6.52745E+02	6.27372E+02	1.72738E+01
WOA	5.87433E+02	6.50855E+02	6.17669E+02	1.56475E+01
<b>Proposed solution</b>	<b>5.84060E+02</b>	<b>5.90580E+02</b>	<b>5.84443E+02</b>	<b>1.58134E+00</b>

**Fig. 38** Convergence rates of the algorithms on the IRIS network



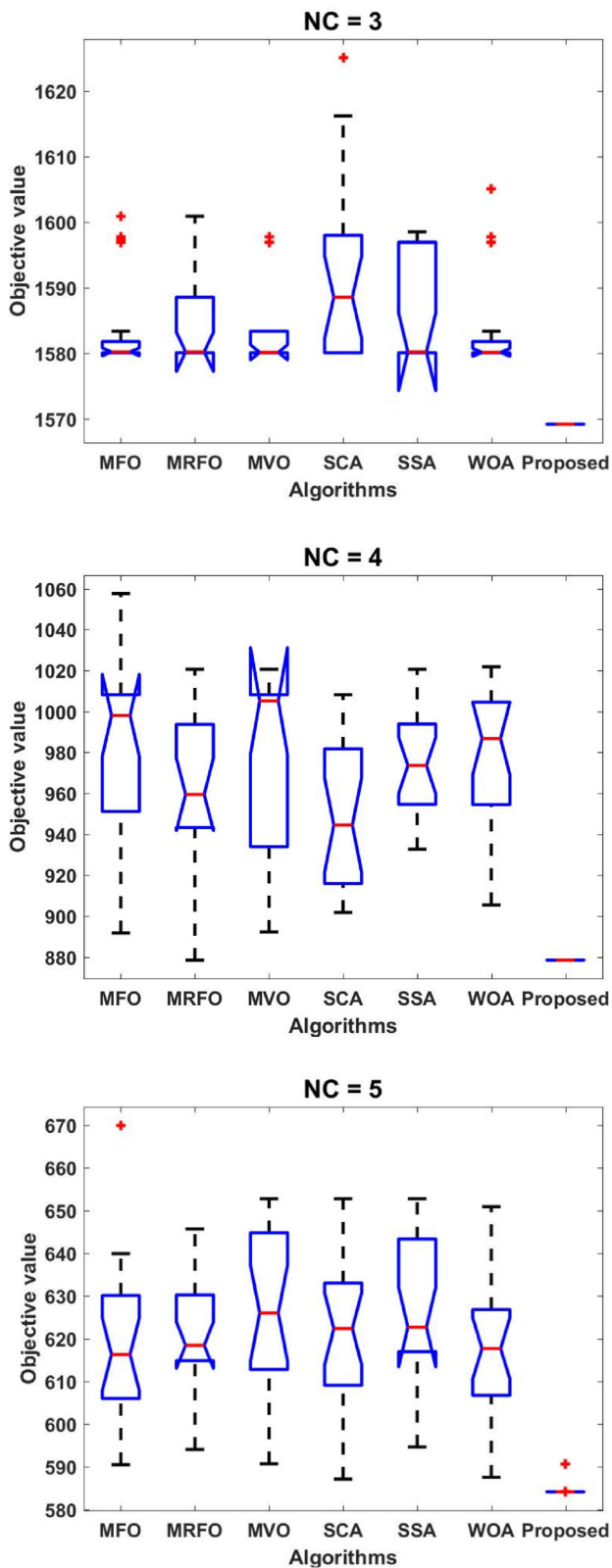


Fig. 39 Box plots of the algorithms on the IRIS network

placement algorithm’s efficiency for the real-world SDN networks with a different number of controllers.

## 6 Conclusion

The present paper addressed the controller placement problem (CCP) in software-defined networks (SDNs) and imparted a novel algorithm called PHCPA. The PHCPA used the network portioning concept and the hybrid Manta-Ray Foraging Optimization algorithm (MRFO) with Salp Swarm Algorithm (SSA). Minimizing network propagation latency was the main objective of the current paper.

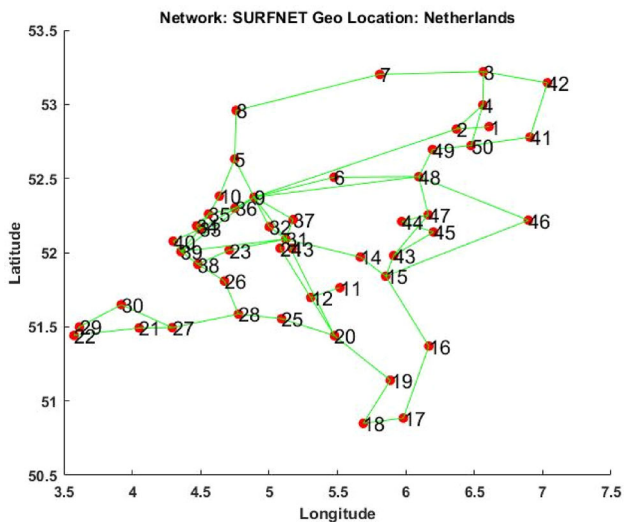
The MRFO and SSA algorithms were powerful nature-inspired meta-heuristic algorithms recently developed to solve various incessant problems. However, given that the CCP is a discrete problem, the MRFO and SSA algorithms were discretized by the two-point swap, random insert, and half points crossover operators. Afterward, they were hybridized to enhance their search capabilities, and their deficiencies such as slow convergence and trapping into local optimums would be eliminated. Henceforth, the resulting discrete hybrid optimization algorithm, was adopted for the controller placement problem. Subsequently, the PHCPA was applied to six real-world software-defined networks embracing topology-zoo to investigate the proposed algorithm’s performance. In the experiments, the PHCPA was compared with state-of-the-art meta-heuristic-based algorithms regarding the best, worst, and average network propagation obtained by the algorithms. Additionally, the results’ standard deviations of the algorithms’ independent runs were compared.

Moreover, for a more in-depth investigation of the algorithms’ performance, the algorithms’ convergence rate was also compared. It is worth mentioning that the experiments on each software-defined network were repeated on three different numbers of controllers for further performance evaluations. The experimental results verified the effectiveness of the contributions and superiority of the proposed controller placement algorithm.

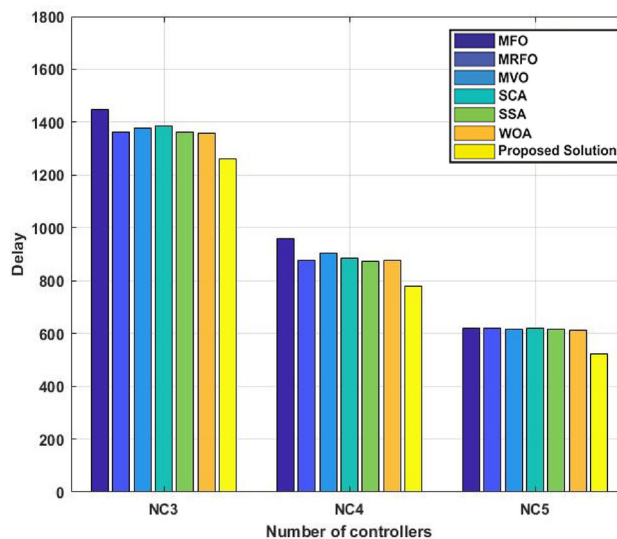
The proposed algorithm had a supreme convergence rate and preminent exploration and exploitation capabilities. Nonetheless, it took more CPU time and required additional storage space. For future research, enthusiasts could recruit other algorithms or apply other metrics along with the propagation latency.

**Table 10** Statistical results of the algorithms obtained on the SURFNET network over 30 independent runs in terms of network delay

	Minimum delay	Maximum delay	Average delay	Standard deviation
NC = 3				
MFO	1.33371E+03	1.44684E+03	1.35117E+03	3.19392E+01
MRFO	1.33371E+03	1.36337E+03	1.34614E+03	1.00880E+01
MVO	1.33371E+03	1.37982E+03	1.35093E+03	1.54867E+01
SCA	1.33371E+03	1.38727E+03	1.34997E+03	1.41367E+01
SSA	1.33371E+03	1.36337E+03	1.34782E+03	8.67835E+00
WOA	1.33371E+03	1.35819E+03	1.34309E+03	1.05775E+01
<b>Proposed solution</b>	<b>1.26047E+03</b>	<b>1.26047E+03</b>	<b>1.26047E+03</b>	<b>0.00000E+00</b>
NC = 4				
MFO	7.96421E+02	9.57431E+02	8.53894E+02	3.13870E+01
MRFO	8.10903E+02	8.76235E+02	8.48505E+02	1.85395E+01
MVO	7.98521E+02	9.03170E+02	8.40918E+02	2.61239E+01
SCA	7.98523E+02	8.85960E+02	8.45893E+02	2.33396E+01
SSA	8.04298E+02	8.73481E+02	8.38339E+02	1.96535E+01
WOA	8.12041E+02	8.74741E+02	8.40571E+02	1.30140E+01
<b>Proposed solution</b>	<b>7.77989E+02</b>	<b>7.77989E+02</b>	<b>7.77989E+02</b>	<b>2.28865E-13</b>
NC = 5				
MFO	5.67905E+02	6.21245E+02	5.93875E+02	1.41006E+01
MRFO	5.81933E+02	6.20279E+02	6.01817E+02	1.09614E+01
MVO	5.82399E+02	6.17876E+02	5.95729E+02	9.47058E+00
SCA	5.54172E+02	6.20303E+02	5.94877E+02	1.60709E+01
SSA	5.75014E+02	6.14520E+02	5.93925E+02	1.05462E+01
WOA	5.67905E+02	6.11856E+02	5.90484E+02	1.09693E+01
<b>Proposed solution</b>	<b>5.23083E+02</b>	<b>5.23083E+02</b>	<b>5.23083E+02</b>	<b>7.82446E-14</b>



**Fig. 40** The SURFNET network



**Fig. 41** Bar graphs of the maximum delay obtained by the algorithms on the SURFNET network



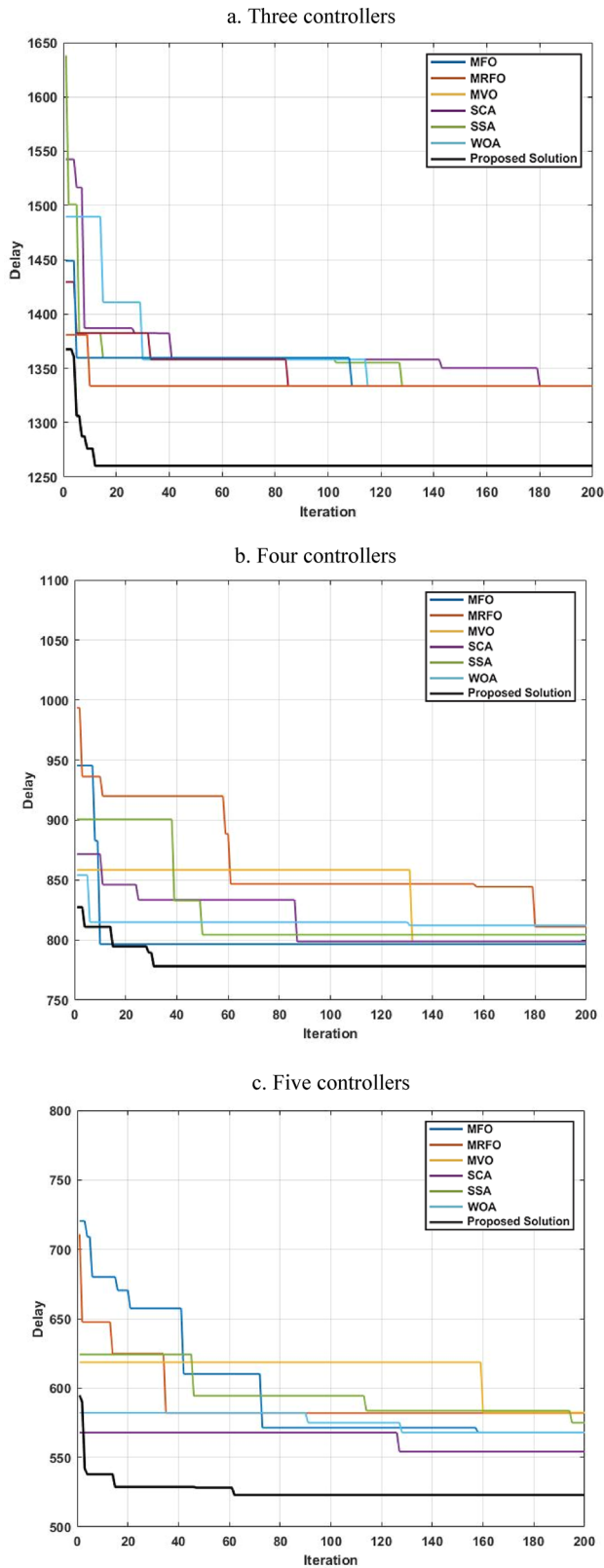


Fig. 42 Convergence rates of the algorithms on the SURFNET network

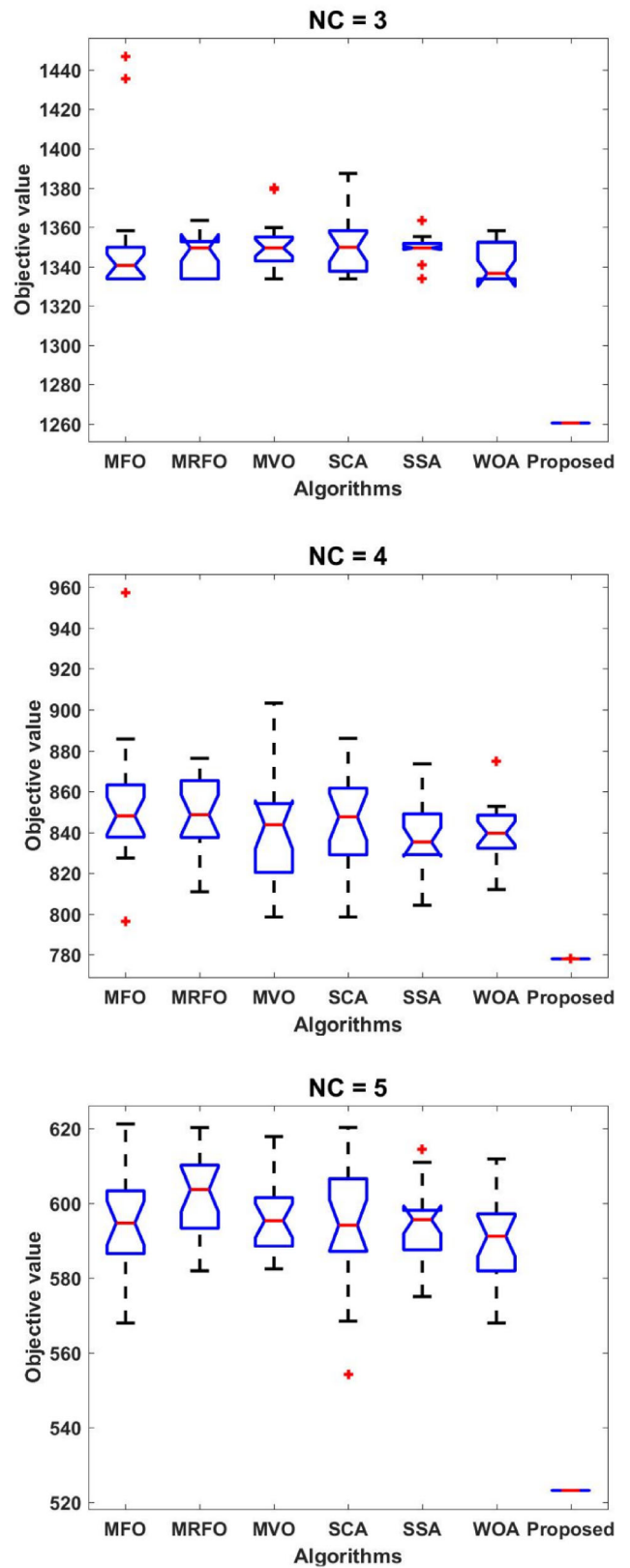


Fig. 43 Box plots of the algorithms on the SURFNET network

## References

- Al-Qerem, A., et al.: IoT transaction processing through cooperative concurrency control on fog–cloud computing environment. *Soft. Comput.* **24**(8), 5695–5711 (2020)
- Masdari, M., et al.: Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. *Cluster Comput.* (2019). <https://doi.org/10.1007/s10586-019-03026-9>
- Masdari, M., Khoshnevis, A.: A survey and classification of the workload forecasting methods in cloud computing. *Cluster Comput.* **23**, 2399–2424 (2019)
- Al-Sharif, Z.A., et al.: Live forensics of software attacks on cyber–physical systems. *Future Gener. Comput. Syst.* **108**, 1217–1229 (2020)
- Iqbal, S., et al.: Minimize the delays in software defined network switch controller communication. *Concurr. Comput.: Pract. Exp.* (2020). <https://doi.org/10.1002/cpe.5940>
- Bhushan, K., Gupta, B.B.: Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment. *J. Ambient Intell. Humaniz. Comput.* **10**(5), 1985–1997 (2019)
- Hu, F., Hao, Q., Bao, K.: A survey on software-defined network and openflow: from concept to implementation. *IEEE Commun. Surv. Tutor.* **16**(4), 2181–2206 (2014)
- Shaghghi, A., et al.: Software-Defined Network (SDN) Data plane security: issues, solutions, and future directions. In: *Handbook of Computer Networks and Cyber Security*, pp. 341–387. Springer, Cham (2020)
- Singh, S., Jha, R.K.: A survey on software defined networking: architecture for next generation network. *J. Netw. Syst. Manag.* **25**(2), 321–374 (2017)
- Rawat, D.B., Reddy, S.R.: Software defined networking architecture, security and energy efficiency: a survey. *IEEE Commun. Surv. Tutor.* **19**(1), 325–346 (2016)
- Moradi, A., Abdi Seyedkolaei, A., Hosseini, S.A.: Controller placement in software defined network using iterated local search. *J. AI Data Min.* **8**(1), 55–65 (2020)
- Abuarqoub, A.: A review of the control plane scalability approaches in software defined networking. *Future Internet* **12**(3), 49 (2020)
- El Kamel, A., Youssef, H.: Improving switch-to-controller assignment with load balancing in multi-controller software defined WAN (SD-WAN). *J. Netw. Syst. Manag.* (2020). <https://doi.org/10.1007/s10922-020-09523-2>
- Jalili, A., Keshtgari, M., Akbari, R.: A new framework for reliable control placement in software-defined networks based on multi-criteria clustering approach. *Soft Comput.* **24**(4), 2897–2916 (2020)
- Singh, A.K., et al.: Heuristic approaches for the reliable SDN controller placement problem. *Trans. Emerg. Telecommun. Technol.* **31**(2), e3761 (2020)
- Fan, Y., Ouyang, T., Yuan, X.: Controller placements for improving flow set-up reliability of software-defined networks. In: *Urban Intelligence and Applications*, pp. 3–13. Springer, Cham (2020)
- Smimesh, C., Kanaga, E.G.M., Sreejish, A.: A multi-controller placement strategy in software defined networks using affinity propagation. *Int. J. Internet Technol. Secured Trans.* **10**(1–2), 229–253 (2020)
- Killi, B.P.R., Rao, S.V.: Poly-stable matching based scalable controller placement with balancing constraints in SDN. *Comput. Commun.* (2020). <https://doi.org/10.1016/j.comcom.2020.02.053>
- Smimesh, C., Grace Mary Kanaga, E., Sreejish, A.: Augmented affinity propagation-based network partitioning for multiple controllers placement in software defined networks. *J. Comput. Theor. Nanosci.* **17**(1), 228–233 (2020)
- ul Huque, M.T.I., Jourjon, G., Gramoli, V.: Revisiting the controller placement problem. In: *2015 IEEE 40th Conference on Local Computer Networks (LCN)*. IEEE (2015)
- Schütz, G., Martins, J.: A comprehensive approach for optimizing controller placement in Software-Defined Networks. *Comput. Commun.* (2020). <https://doi.org/10.1016/j.comcom.2020.05.008>
- Heller, B., Sherwood, R., McKeown, N.: The controller placement problem. *ACM SIGCOMM Comput. Commun. Rev.* **42**(4), 473–478 (2012)
- Muluye, W.: A review on software-defined networking distributed controllers. *Int. J. Eng. Comput. Sci.* **9**(2), 24953–24961 (2020)
- Yao, Z., Yan, Z.: A trust management framework for software-defined network applications. *Concurr. Comput.: Pract. Exp.* **32**(16), e4518 (2020)
- Barshandeh, S., Piri, F., Sangani, S.R.: HMPA: an innovative hybrid multi-population algorithm based on artificial ecosystem-based and Harris Hawks optimization algorithms for engineering problems. *Eng. Comput.* (2020). <https://doi.org/10.1007/s00366-020-01120-w>
- Barshandeh, S., Haghzadeh, M.: A new hybrid chaotic atom search optimization based on tree-seed algorithm and Levy flight for solving optimization problems. *Eng. Comput.* (2020). <https://doi.org/10.1007/s00366-020-00994-0>
- Masdari, M., Barshandeh, S., Ozdemir, S.: CDABC: chaotic discrete artificial bee colony algorithm for multi-level clustering in large-scale WSNs. *J. Supercomput.* **75**(11), 7174–7208 (2019)
- Masdari, M., Barshandeh, S.: Discrete teaching–learning-based optimization algorithm for clustering in wireless sensor networks. *J. Ambient Intell. Humaniz. Comput.* (2020). <https://doi.org/10.1007/s12652-020-01902-6>
- Faramarzi, A., et al.: Equilibrium optimizer: a novel optimization algorithm. *Knowl.-Based Syst.* **191**, 105190 (2020)
- Hayyolalam, V., Kazem, A.A.P.: Black widow optimization algorithm: a novel meta-heuristic approach for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **87**, 103249 (2020)
- Dhiman, G., Kumar, V.: Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems. *Knowl.-Based Syst.* **165**, 169–196 (2019)
- Brammya, G., et al.: Deer hunting optimization algorithm: a new nature-inspired meta-heuristic paradigm. *Comput. J.* (2019). <https://doi.org/10.1093/comjnl/bxy133>
- Kaveh, A., Zaeerza, A.: Shuffled shepherd optimization method: a new meta-heuristic algorithm. *Eng. Comput.* (2020). <https://doi.org/10.1108/EC-10-2019-0481>
- Sulaiman, M.H., et al.: Barnacles Mating Optimizer: a new bio-inspired algorithm for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **87**, 103330 (2020)
- Mohanty, S., et al.: A simulated annealing strategy for reliable controller placement in software defined networks. In: *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE (2020)
- Abuabara, R.I., et al.: Cost-effective Tabu search algorithm for solving the controller placement problem in SDN. In: *Pattern Recognition Applications in Engineering*, pp. 109–130. IGI Global, Hershey (2020)
- Griffin, L., Zuccarelli, L.: Software defined network optimization using quantum computing. Google Patents 2020
- Li, Y., Sun, W., Guan, S.: A multi-controller deployment method based on PSO algorithm in SDN environment. In: *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE (2020)

39. Akbar Neghabi, A., et al.: Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network. *Int. J. Commun. Syst.* **32**(4), e3875 (2019)
40. Liao, W.-C., et al.: System and method for joint power allocation and routing for software defined networks. Google Patents 2019
41. Xu, Y., et al.: Dynamic switch migration in distributed software-defined networks to achieve controller load balance. *IEEE J. Sel. Areas Commun.* **37**(3), 515–529 (2019)
42. Ateya, A.A., et al.: Chaotic salp swarm algorithm for SDN multi-controller networks. *Eng. Sci. Technol.* **22**(4), 1001–1012 (2019)
43. Gao, C., et al.: A particle swarm optimization algorithm for controller placement problem in software defined network. In: *International Conference on Algorithms and Architectures for Parallel Processing*. Springer (2015)
44. Hu, Y., et al.: The energy-aware controller placement problem in software defined networks. *IEEE Commun. Lett.* **21**(4), 741–744 (2016)
45. Liyanage, K.S.K., Ma, M., Chong, P.H.J.: Controller placement optimization in hierarchical distributed software defined vehicular networks. *Comput. Netw.* **135**, 226–239 (2018)
46. Singh, A.K., Maurya, S., Srivastava, S.: Varna-based optimization: a novel method for capacitated controller placement problem in SDN. *Front. Comput. Sci.* **14**(3), 143402 (2020)
47. Wang, G., et al.: An effective approach to controller placement in software defined wide area networks. *IEEE Trans. Netw. Serv. Manag.* **15**(1), 344–355 (2017)
48. Wang, H., et al.: Load-balancing routing in software defined networks with multiple controllers. *Comput. Netw.* **141**, 82–91 (2018)
49. Kanodia, K., et al.: CCPGWO: A meta-heuristic strategy for link failure aware placement of controller in SDN. In: *2020 International Conference on Inventive Computation Technologies (ICICT)*. IEEE (2020)
50. Kanodia, K., et al.: HPSOSA: a hybrid approach in resilient controller placement in SDN. In: *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*. IEEE (2020)
51. Li, Y., Sun, W., Guan, S.: A firefly inspired controller placement algorithm in software defined network. In: *2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET)*. IEEE (2019)
52. Sahoo, K.S., et al.: On the placement of controllers in software-defined-WAN using meta-heuristic approach. *J. Syst. Softw.* **145**, 180–194 (2018)
53. Jalili, A., Keshtgari, M., Akbari, R.: Optimal controller placement in large scale software defined networks based on modified NSGA-II. *Appl. Intell.* **48**(9), 2809–2823 (2018)
54. Tahmasebi, S., et al.: Cuckoo-PC: an evolutionary synchronization-aware placement of SDN controllers for optimizing the network performance in WSNs. *Sensors* **20**(11), 3231 (2020)
55. Tootoonchian, A., et al.: On controller performance in software-defined networks. In: *2nd {USENIX} Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 12)* (2012)
56. Nasiri, A.A., Derakhshan, F.: Assignment of virtual networks to substrate network for software defined networks. *Int. J. Cloud Appl. Comput. (IJCAC)* **8**(4), 29–48 (2018)
57. Zhao, W., Zhang, Z., Wang, L.: Manta ray foraging optimization: an effective bio-inspired optimizer for engineering applications. *Eng. Appl. Artif. Intell.* **87**, 103300 (2020)
58. Mirjalili, S., Lewis, A.: The whale optimization algorithm. *Adv. Eng. Softw.* **95**, 51–67 (2016)
59. Mirjalili, S., et al.: Salp Swarm Algorithm: a bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **114**, 163–191 (2017)
60. Masdari, M., et al.: Optimization of airfoil Based Savonius wind turbine using coupled discrete vortex method and salp swarm algorithm. *J. Clean. Prod.* **222**, 47–56 (2019)
61. El-Ashmawi, W.H., Ali, A.F.: A modified salp swarm algorithm for task assignment problem. *Appl. Soft Comput.* (2020). <https://doi.org/10.1016/j.asoc.2020.106445>
62. Xia, C., et al.: Path planning and energy flow control of wireless power transfer for sensor nodes in wireless sensor networks. *Turk. J. Electr. Eng. Comput. Sci.* **26**(5), 2618–2632 (2018)
63. Robusto, C.C.: The cosine-haversine formula. *Am. Math. Mon.* **64**(1), 38–40 (1957)
64. Sierpinski, W.: *Pythagorean Triangles*, vol. 9. Courier Corporation, Chelmsford (2003)
65. Weisstein, E.W.: *Pythagorean Theorem*. <https://mathworld.wolfram.com/> (2006)
66. Johnson, R.: *Spherical Trigonometry*. West Hills Institute of Mathematics. [https://www.math.ucla.edu/~robjohn/math/sphere\\_trig.pdf](https://www.math.ucla.edu/~robjohn/math/sphere_trig.pdf)
67. Liao, J., et al.: Density cluster based approach for controller placement problem in large-scale software defined networkings. *Comput. Netw.* **112**, 24–35 (2017)
68. Lange, S., et al.: Heuristic approaches to the controller placement problem in large scale SDN networks. *IEEE Trans. Netw. Serv. Manag.* **12**(1), 4–17 (2015)
69. Yoon, S.-K., et al.: Controller placement algorithms in software defined network—a review of trends and challenges. In: *MATEC Web of Conferences*. EDP Sciences (2017)
70. Qi, Y., et al.: Towards multi-controller placement for SDN based on density peaks clustering. In: *ICC 2019- 019 IEEE International Conference on Communications (ICC)*. IEEE (2019)
71. Wang, G., et al.: A K-means-based network partition algorithm for controller placement in software defined network. In: *2016 IEEE International Conference on Communications (ICC)*. IEEE (2016)
72. Masdari, M., Khezri, H.: Efficient VM migrations using forecasting techniques in cloud computing: a comprehensive review. *Cluster Comput.* (2020). <https://doi.org/10.1007/s10586-019-03032-x>
73. Mirjalili, S.: Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **89**, 228–249 (2015)
74. Mirjalili, S., Mirjalili, S.M., Hatamlou, A.: Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **27**(2), 495–513 (2016)
75. Mirjalili, S.: SCA: a sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **96**, 120–133 (2016)

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



**Nasrin Firouz** was born in 1983, obtained her B.Eng. degree in Computer Software Engineering from Islamic Azad University, Khoy Branch, Iran, in 2008. She has received her M.Eng. degree in Computer Architecture Engineering from Islamic Azad University, Tabriz Branch, Iran, in 2011. Currently, she is working toward the Ph.D. degree in Software systems Engineering at the Islamic Azad University, Urmia branch, Iran. Her research is focused on

software defined network, optimization and network virtualization.



**Amin Babazadeh Sangar** finished PhD of Information Systems in 2014 at Universiti Teknologi Malaysia (Leading University in Innovation, Entrepreneurship and Technology). His PhD research is mainly focused on Business Intelligence Systems and through a qualitative research on case study companies he reached a useful method for measuring success of BI systems. During my PhD he awarded Fellowship and Scholarship

from Malaysian Ministry of Higher educations.



**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 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.



**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, Azerbaijan in 2005 and 2009, respectively. His research interests are very large-scale integration design and computer networking.