Hindawi
Journal of Computer Networks and Communications
Volume 2022, Article ID 3145276, 18 pages
https://doi.org/10.1155/2022/3145276

Research Article

Multi-Controller Placement Optimization Using Naked Mole-Rat Algorithm over Software-Defined Networking Environment

A. Binod Sapkota, B. Babu R. Dawadi, and C. Shashidhar R. Joshi

Department of Electronics and Computer Engineering, Institute of Engineering, Tribhuvan University, Kirtipur, Nepal

Correspondence should be addressed to C. Shashidhar R. Joshi; srjoshi@ioe.edu.np

Received 16 August 2022; Revised 10 November 2022; Accepted 18 November 2022; Published 13 December 2022

Academic Editor: Giovanni Pau

Software Defined Networking (SDN) is the novel networking paradigm where decoupling of the control plane from the data plane has its inherent advantages. Controller Placement Problem (CPP) involves placing the optimal number of controllers at the appropriate locations while meeting prerequisites such as latency, load balancing, energy and computational time. To achieve scalability, deployment of multiple controllers on large-scale SDN is one of the key challenges. CPP can be addressed as a multi-objective combinatorial optimization problem whose solution is a trade-off between multiple optimization parameters. In this paper, a novel population-based meta-heuristic algorithm viz. Naked Mole-Rat (NMR) Algorithm has been proposed to optimize the location for controller placement based on Switch-Controller (SC), Controller-Controller (CC) latency while maintaining load balancing among the controllers. The ideas and mechanisms are illustrated using two publicly available standard topologies viz. Ernet and Savvis. The controller localization approach implemented with NMR algorithm has slightly a better result as compared with the Bat algorithm.

1. Introduction

1.1. Software-Defined Networking (SDN). SDN is a new paradigm that enables the dynamic nature of networks and smart services while offering simple network management allowing consumers to have a higher quality of experience while lowering expenses during its implementation, operation and maintenance [1].

In SDN, the network control plane is separated from the data plane and moved to a distinct entity known as the SDN controller. The close coupling of the control plane, which chooses how to manage network traffic, and the forwarding plane, which delivers traffic based on the control plane's choices, restrict the flexibility and advancement on traditional networks. The network in SDN is programmable by software applications that run on top of the Network Operating System (NOS), and forwarding decisions are made depending on traffic. One of the most significant advantages in SDN is that network intelligence is centralized in software-based controllers [2]. SDN transforms the legacy switches into pure forwarding elements whose flow tables are populated by the controller.

1.2. Controller Placement Problem (CPP). The location of controllers in a distributed SDN controller architecture has a significant impact on network performance in terms of latency, dependability, and other factors, which is known as the CPP [3]. The number of controllers to be deployed and their placement is two facets of the CPP.

The number of controllers that must be assigned to switches and their right location in SDN is a critical factor in attaining a faster flow setup time with improved fault management and controller load balancing [4].

Performance and deployment costs are traded-off when determining the number of controllers needed [2]. Controller placement is also a problem in an SDN with a single physical controller, but the problem is less pronounced. Because of their uneven topology and considerable packet

propagation delay, controller placement is especially important in wide-area networks [5].

As the number of controllers and their location of deployment has significantly impacted the performance of the network in multi-controller network architecture, the CPP has been a research hot-spot. Thus, in order to achieve several defined goals, such as latency minimization, load balancing, energy efficiency, and enhanced reliability, which is crucial to SDN's performance in large-scale networks, CPP seeks to determine the best position for the SDN controllers [6].

The CPP typically relates to the number and location of controllers in a network [7]. The CPP study was initially conducted by Heller et al. [8]. The problem was reformulated as a facility location problem and shown to be NP-hard which normally regards the quantity and location of controllers in network. There have been countless attempts since then to position the controllers in the best possible location.

To solve the NP-hard problem, efficient meta-heuristic algorithms such as Particle Swarm Optimization (PSO), FireFly Algorithm (FFA), Varna Based Optimization (VBO), *k*-means, Grey Wolf Optimization (GWO), Bacterial Foraging Optimization (BFO) and so on for latency measurements criteria have been studied and designed for different networks [9].

Salgotra and Singh [10] formulate the swarm intelligent nature-inspired algorithm called Naked Mol-Rat NMR algorithm and its performance is evaluated by a comparative study with other algorithms. Their experimental results and statistical analysis prove that NMR algorithm is very competitive as compared to other advanced algorithms like PSO, GWO, Whale Optimization Algorithm (WOA), Differential Evolution (DE), Gravitational Search Algorithm (GSA), Fast Evolutionary Programming (FEP), Bat Algorithm, Fast Pollination Algorithm (FPA), and FFA.

Although Nature-inspired algorithms may have certain drawbacks, more and more optimization issues are now being solved using nature-inspired algorithms to find the best solution. Additionally, because of their simple conceptual model and little requirement of gradient data, they are also easy to construct. Hence, We are highly motivated to choose this NMR algorithm to solve the CPP in SDN. Therefore, in this paper, NMR algorithm, a novel meta-heuristic swarm intelligence algorithm has been implemented to solve the optimization problem of controller placement.

Numerical measures such as latency, load imbalance, energy, computation time, cost, connectivity, and control plane overhead, among others, have been presented in the literature to address the controller placement problem in SDN. Switch to controller (SC) and controller to controller (CC) latency play the most important roles among these measures since they have a significant impact on SDN's overall performance [11]. Therefore, in this study, we evaluate latency and look for ways to minimize it so that controller placement in wide-area networks can be optimized. The main contributions of this paper are briefly highlights as follows.

(1) NMR algorithm for optimum controller placement is proposed.

(2) Proposed NMR algorithm is evaluated with different latency values also considering optimum load balancing and compared the result with Bat algorithm.

(3) Three aspects: (i) effect on average SC latency (ii) effect on global latency (iii) effect of algorithm execution complexity with increase in controllers are included in the analysis to find the optimal solution.

The rest of the paper is organized as follows. Section 2 presents the literature review on controller placement implementation in SDN with a detailed concept of the NMR algorithm. In Section 3, the methodology of our research work including the experimental environment setup and dataset used are discussed. Section 4 presents the overall analysis of our implementation approach and its comparison with others, while Section 5 concludes the paper with possible future works.

2. Literature Review

2.1. Multiple Controllers in SDN. SDN can be logically centralized using a single controller or multiple controllers that share part of their local network information to generate a global view. The single centralized controller cannot keep up with the growing need for flow processing as the network grows in size. Also, a physically centralized controller has significant drawbacks, including the potential for a single point of failure throughout the whole network [12]. As a result, the multi-controller is a potential option for SDN in large-scale networks [13].

The main concept behind using numerous controllers is to evenly share the load throughout the network. Furthermore, when one controller fails, another should take over, resolving both the scalability and robustness challenges [2]. Because the controllers are in charge of producing forwarding rules and filling them onto the switches, the arrangement of the controllers has a significant impact on the network's performance. Multiple controllers are necessary not just to increase network speed, but also to ensure that the network is always available.

Hu et al. [13] suggest multi-controller architecture based on two fundamental multi-controller designs: flat and hierarchical. It is designed to address the shortcomings of single controllers, such as single controller failure and restricted controller capacity.

Multi-controller, on the other hand, introduces scalability issues, such as how to choose controller locations and how to assign switches for multi-controller in the network. In reality, the scalability of a multi-controller system is determined by the number of controllers and deployment strategy. If controllers are deployed arbitrarily, it may result in an imbalanced processing load on controllers, reducing control plane capacity. Therefore, for the large-scale SDN networks to achieve scalability, fault tolerance and decreased latency, the deployment of several controllers and cooperative work among them is the must.

The network's servers get overloaded as the number of requests from users grow. As a result, the load must be balanced to deliver better service and meet Quality of Service (QoS) standards. The connection will fail and the server will crash if all of these issues are ignored. The objective of a load balancer is to maximize bandwidth to increase efficiency while maintaining low latency, effectively using resources without deadlocks, and without contributing to the network's overhead. However, there are issues with reliability and scalability when using a single controller. As a result, for east-west interfaces, obtaining distributed multiple controllers to address this problem is an alternative that allows those controllers to connect [14].

Load balancing has a significant influence on SDN performance and availability. As a result, in SDN, the CPP has an impact on load balancing solutions. SDN load balancing divides the data plane from the physical network control plane. The management of several devices is possible using an SDN-based load balancer. Networks can become more agile in this way. Direct programming of the network control can result in application services that are more responsive and effective. Networks have lagged behind in terms of virtualization and automation while computing and storage have witnessed advancements. The network can perform like the virtualized versions of computation and storage thanks to load balancing using SDN.

2.2. Naked Mole-Rat (NMR) Algorithm. NMR algorithm [10] is a stochastic optimization algorithm. Based on the social behavior of the NMR, which imitates the mating pattern of these animals. The key features are summarized as follows.

(1) NMR, a eusocial animal, can have upto 295 members with average number of members being 70–80.

(2) The group is led by a female queen and the entire population is divided into breeders and workers. The most efficient NMRs among working groups are breeders and are intended for mating with the queen.

(3) All other necessary tasks are performed by the workers and the most efficient worker will be promoted to breeder's group. Thus, the workers who outperform the other workers will become breeders while the breeders who perform the worst will be shifted back to the worker's pool.

(4) Ultimately, only the best breeder among the breeders will be the queen's mating partner.

These above mentioned rules formularize the concept of Naked Mole-Rat Algorithm (NMRA). The population of NMR is initialized during the first phase. Then the NMR population is divided into workers and breeders. The selection of the breeders is based on their breeding probability. The algorithm can be described in steps listed below.

(1) *Initialization*: The population of n NMRs are randomly generated in the range of $[1, 2, \ldots, n]$. Each NMR is represented in D-dimensional vector space. Here, D represents the number of variables or parameters to be tested in the problem. Each NMR is initialized as give in.

$$\mathrm{NMR}_{ij} = \mathrm{NMR}_{\min.j} + U(0,1) \times \left(\mathrm{NMR}_{\min.j} - \mathrm{NMR}_{\max.j}\right), \tag{1}$$

where $i \in [1, 2, \ldots, n]$, $j \in [1, 2, \ldots, d]$, $\mathrm{NMR}_{i.j}$ is the i^{th} solution in the j^{th} dimension, $\mathrm{NMR}_{\min.j}$, $\mathrm{NMR}_{\max.j}$ are the lower and upper bounds of the problem function respectively and $U(0,1)$ is the uniformly distributed random number. The objective function is evaluated and its fitness is estimated after it has been initialized. b breeders and w workers are determined based on fitness, and the overall initial best solution d is calculated. After initialization, the NMR population is subjected to multiple cycles or iterations of the worker and breeder phases of the search process.

(2) *Worker phase*: During this phase, the workers are inclined to improve fitness to gain opportunity to become a breeder and eventually mate with the queen. The new solution based on its previous experience and its information available locally. Old solution is memorized unless the mating fitness is better than the previous one. The best fitness of worker will be remembered after the completion of search process. A new solution is produced from old solution given by:

$$w_i^{t+1} = w_i^t + \lambda\left(w_j^t - w_k^t\right), \tag{2}$$

where w_i^t corresponds to the i^{th} worker in the t^{th} iteration, w_i^{t+1} is the new solution or the worker, λ is the mating factor and w_j^t and w_k^t are two random solution chosen from the worker's pool. The value of λ is obtained from the uniform distribution in the range of [0, 1].

(3) *Breeder phase*: In breeder phase, the breeder NMR updates the fitness to retain its position as a breeder and improve its chances of mating. All the breeders are updated based on its breeding probability (BP) with the best fitness as their baseline. The value of BP lies in the range [0, 1]. If a breeder can't update its fitness, as its breeding probability is low, then it could be pushed back to worker category. The positional update is based on.

$$b_i^{t+1} = (1-\lambda) b_t^i + \lambda\left(d - b_i^t\right). \tag{3}$$

Here, b_i^t corresponds to the breeder i in the iteration t, λ factor controls the mating frequency of breeders and helps in identifying a new breeder b_i^{t+1} in the next iteration.

The concept of mating pattern of breeders with the queen is used for finding the appropriate solution of the considered problem. On the basis of the idea that the best workers can be promoted to the breeders and the best breeders drift towards mating the queen is followed to devise a new solution. The algorithm focuses on determining the best breeder which can mate with queen and thus devising the potential best solution of the problem. Breeding probability controls the shifting of worker phase towards the breeding phase. The two random worker mole-rats in the close proximity to each other control the worker phase while the best mole-rat and some random breeder in close proximity of the current best solution. A simple random scaling factor, known as mating factor, governs the breeder and workers. In terms of local and global search, the worker phase of NMRA corresponds to exploration operation while the breeder phase correlates to the exploitation operation. The exploration operation is governed by two random solutions, while exploitation operation has one random solution, which is close to the current best solution [15].

2.3. Related Work. The SDN CPP investigates the number and position of SDN controllers required to fulfil control plane needs in a pure SDN network. It was initially proposed by Heller et al. [8] to reduce control channel latency, and it has been given a lot of attention for objectives like resiliency, dependability, fault tolerance, survival, energy efficiency, load balancing, and many more.

Fan et al. [3] proposed a unique multiobjective metaheuristic-based Reliability-Aware and Latency-Oriented (RALO) controller placement algorithm. They tested it on eight real networks and two generated networks conforming to the Erdos–Renyi (ER) random model and the small-world model. The simulation results revealed that the proposed technique might provide competitive switch-to-controller latencies in both no-link and single-link failure scenarios.

During the real-time migration of an existing legacy network into a Software Defined IPv6 (SoDIP6) network, the appropriate location of the SDN control plane is determined by evaluating the shortest control route latency utilizing optimum path routing and the Breadth-First Router Replacement (BFR) approach [4]. According to their simulation results, the BFR strategy for controller placement and router migrations outperforms sequential router migrations in the best path. The number of controllers required and their location in SDN has become key challenges as more routers transition to SDN switches. Much of the research on controller location has been linked to pure SDN. In the progressive implementation of SoDIP6 networks across hybrid SDN/legacy networks, they examined master controller location.

Wang et al. [5] explore and investigate possible contributors to the end-to-end latency and the queuing latency of controllers. To decrease the end-to-end latency, the concept of a network partition is introduced and a Clustering-based Network Partition Algorithm (CNPA) is then proposed to partition the network. The CNPA can guarantee that each partition can shorten the maximum end-to-end latency between controllers and switches. To further decrease the queuing latency of controllers, appropriate multiple controllers are then placed in the sub-networks. Extensive simulations are carried out under two real network topologies. The results verify that the proposed algorithm can remarkably reduce the maximum latency between controllers and their associated switches.

Hu et al. [13] conclude from their review of the literature that the present research addresses the problem in two ways: (1) controller clustering and (2) switch migration. In comparison, controller clustering focuses on an architectural design by building a dynamic controller resource pool, whereas switch migration focuses on modifying controller load distribution to maintain load balancing.

Babbar et al. [14] presented a highly scalable load balancing technique based on delay. Their suggested solution tackles load-balancing challenges with numerous overloaded controllers and migrates a load of overloaded switches to other controllers in a short time on the SDN control plane by finding the right latency and resolving several overloads concurrently. In addition to the migration, their technique reduced latency by 25% as compared with the previous solutions. As a result, the suggested technique enables efficient and timely load balancing of numerous controllers in SDN.

Rasol Domingo Pascual [16] proposed the Control Plane Latency (CPL) metric to evaluate how good the Joint Latency and Reliability-Aware Controller Placement (LRCP) placements are in a real controller deployment.

Wang et al. [17] explored a CNPA approach and investigated the entire delay between controllers and switches. To evaluate the performance of the proposed technique, extensive simulations were done using two real-world topologies obtained from the Internet Topology Zoo. The CNPA was shown to successfully reduce overall latency in simulations when many controllers are deployed into each sub-network and compared to K-means and K-center.

Cui et al. [18] proposed a novel SDN multiple controller load-balancing technique based on (real-time) reaction time i.e. Strategy of Multiple SDN Controllers Based on Response Time (SMCLBRT). In the migration decision-making process, SMCLBRT always chooses the switch that has the greatest influence on the master controller's migration response time. The simulations show that their strategy can start migration ahead of schedule and lessen the workload on overloaded controllers immediately.

By considering the expected sets of major targeted attacks on network architecture, Calle et al. [19] presented an optimization technique that includes an algorithm for predicting the most dangerous attack sets to properly deploy controllers. The controller placement optimization is then carried out using mixed integer programming techniques using the data from these sets as input. Investigations into additional backup controllers are being done to decrease the impact of attacks.

Using four parameters: latency, reliability, cost, and numerous objective optimizations, Lu et al. [20] offered alternative possibilities for controller placement and load balancing in SDN. The absence of suitable controller

placement in dynamic traffic situations might cause network delays. The method is divided into two parts: optimizing controller location and optimizing controller devotion time to switches under various traffic scenarios.

Liao et al. [21] also proposed a Density-Based Controller Placement (DBCP) switch clustering algorithm to split the network into several sub-networks based on the network architecture, and the optimal number of controllers is obtained according to DBCP switch clustering. By incorporating clusters, DBCP considers latency, load balancing, and link failures in real-world networks. They have shown the improved performance of experimental results and are simply transferable to actual networks.

CPP for Software-Defined Wireless Sensor Networking (SDWSN) and controller replacement in the event of SDWSN failure were proposed by Kobo et al. [22]. To assure the least amount of latency and resiliency, they optimized their proposed fragmentation model for SDWSN using k-means and k-center and integrating CPP and an effective controller re-election method. They consider propagation latency, the optimal number of controllers to deploy, and failure resilience. For local controller placement, the k-means was employed, and for global controller placement, the k-center.

Singh et al. [23] also proposed an efficient CPP approach for SDN-based Wide Area Network (WAN) that is heuristic in nature and decreases the total average latency of the SDN network between switches and controllers (SC) as well as between the controllers (CC) to maximize SDN performance. Authors developed a new optimization algorithm known as VBO to solve CPP where switches and controllers are represented as particles. Varna class is not dictated by birth but by particles' fitness value called Karma [24]. The approach considered capacitated, incapacitated, load-aware capacitated, load-aware incapacitated, and latency for controller placement.

An optimization approach for controller placement in SDN was presented by Liao et al. [25]. The multi-objective genetic algorithm Multi-Objective Genetic Algorithm (MOGA) and particle swarm optimization PSO techniques are used in this study to arrange the controllers. This PSO selects a global best position for a particle by considering its delay limitations and the controllers' position and velocity are updated using the genetic algorithm. The genetic algorithm is constrained, nevertheless, by its high computational time consumption.

Radam et al. [26] try to minimize the latency between controllers, minimize the delay between switches, and maximize the controller fault tolerance rate. Initially, the proposed SDN network is constructed based on graph theory to increase the scalability, connectivity, and flexibility of the network, which increases the communication efficiency and reduces the propagation delay of the link. Then, the optimal controller selection is performed by using the FFA, which improves the performance of controller placement to manage the network. Finally, the multi-controller placement is performed by using a hybrid of Harmony Search Algorithm (HSA) and PSO, which reduces the communication latency between the switch and the controller by selecting an optimal location to place the controller. The simulation of multi-controller placement is carried out by the CloudsimSDN network simulator and the simulation results demonstrate good performance.

Khorramizadeh and Ahmadi [27] proposed the SDN CPP as a location-allocation model and the formulated framework solves focusing it in two phases: (i) determining the required number of controllers while minimizing the total cost, and (ii) balancing the controller load with their introduced fair load distribution function and to reduce inter-controller latency. Two greedy procedures are designed for the proposed framework algorithms to solve the models and numerical results show their efficiency.

Guan et al. [28] established the CPP model to decrease the synthetical delay and balance the load of the controllers. They designed the CPP solving algorithm based on the improved FFA and verified its effectiveness by comparison with the latest controller placement algorithms. Experimental results in the real large-scale SDN topologies showed that the controller placement scheme obtained by the improved FFA had a lower synthetical delay and a more balanced controller load. Similarly, the time consumption of the improved FFA is found to be acceptable.

To address the CPP in a distributed 5G network, Ibrahim et al. [29] proposed an efficient, heuristic multi-objective optimization approach using the Dynamic Capacitated Controller Placement Problem (DCCP) based on the K-center problem. They used Greedy Random Search (GRS) to solve the dynamic assignment of nodes to controllers to achieve load balancing. Their claim is that the design of the heuristic method provides proper load balancing, efficient cost management, and network resource management, as compared to the basic Capacitated Controller Placement Problem (CCPP) model.

Gao et al. [30] addressed the CPP utilizing PSO using the SC and CC latency as the evaluation metrics. Through the use of the controller capacity, the author also addressed the idea of load balancing. PSO performance was compared to that of the greedy algorithm and Integer Linear Programming (ILP). Compared to greedy algorithm and ILP, PSO converged more quickly and produced the best results for the larger network.

Li et al. [31] presented the improved FFA to solve the CPP in multi-controller environment. The parameters taken into consideration for optimization are the average latency between SC and controller load usage. The technique is restricted to small networks, and the issue of communication across domains is left unsolved.

Dhar et al. [32] proposed a mathematical algorithm to form the clusters and placed one controller in each cluster to shorten the worst-case SC latency. In result, their proposed technique "$-method" performs better compared with other existing algorithms in terms of worst-case SC latency minimization with less number of controllers. By assigning the switches from a failed controller to the controllers closest to it, they have also studied the failure mode of their method, which demonstrates that it also performs better in terms of network fault tolerance and boosts network resilience.

Salgotra and Singh [10] formulate the swarm intelligent nature-inspired algorithm called NMR algorithm and its performance is evaluated by a comparative study with other algorithms. Their experimental results and statistical analysis prove that NMR algorithm is very competitive as compared to other advanced algorithms like PSO, GWO, WOA, DE, GSA, FEP, Bat algorithm, FPA, and FFA.

The CPP is solved using metaheuristic algorithms, which cover a wide range of difficulties. One of the most recent meta-heuristic algorithms based on the NMRs' mating behavior is the NMRA. When compared to other cutting-edge metaheuristic algorithms, the NMR's performance on the benchmark test functions showed its usefulness and gave it a competitive advantage.

3. Methodology

3.1. Problem Formulation. A network topology is represented by an undirected graph $G(S, E)$, where S represents the set of switches and E represents the set of bidirectional physical links interconnecting the switches. Table 1 provides the set of notations to represent sets, decision variables and constraints.

Average SC latency is the most commonly used metric in CPP. It calculates the average distance between the placed controllers' location and the switches assigned to them. It reflects the basic performance of propagation latency in the SDN by representing the average value of packet transmission latency between the switch and the controller. The mathematical expression can be expressed in

$$L_{sc-\text{avg}} = \frac{1}{n} \sum_{s \in S, c \in C} \min [d(s, c)]. \tag{4}$$

The sub-network created among the controllers' is located on the controller plane, thus the latencies among these controllers is also critical aspect of SDN. Communications between the controllers are crucial for achieving the consistent view of the network's state which is utilized by network application for proper operation. The communication overhead maintained by the shared state among the controller is very significant. The controller to controller latency is mathematically represented by

$$L_{cc-\text{avg}} = \frac{1}{k} \sum_{c \in C, c' \in C} \min [d(c, c')]. \tag{5}$$

It is considered that all the switches are connected to the closest controller based on latency as the metric, unless the addition of the switch exceeds the controller capacity, i.e. the controller has already the maximum number of the switches it can handle. In such scenario, the switch is connected to the second nearest controller. Taking the reference from [33], we assumed a number of OpenFlow requests (γ) in the range of 0.05–0.105 million request per second by an SDN switch, the request-handling capacity of controller (σ) is 1.1 million requests per second. This can be mathematically represented in

$$\sum_{s \in T(c)} l(s) \leq L(c) \quad \forall c \in C. \tag{6}$$

For the placements of C controllers, the global average latency can be represented by

$$G(C) = w_1 \times \frac{1}{n} \sum_{s \in S, c \in C} \min [d(s, c)] + w_2 \times \frac{1}{k} \sum_{c \in C, c' \in C} \min \left[d\left(c_i, c_j\right) \right], \tag{7}$$

where, w_1 and w_2 are weights, such that $w_1 + w_2 = 1$.

The objective function of capacitated controller placement problem for global average latency is given by.

$$\text{Minimize } G(C), \text{ subject to } \sum_{s \in T(c)} l(v) \leq L(C) \quad \forall c \in C. \tag{8}$$

For k number of controllers to be deployed in an SDN, the goal is to optimize the placement of controllers such that the value of $G(C)$ is minimized subjected to the constraint presented in equation (8) and $|C| = k$.

The great circle distance between the pairs of switch is computed using Haversine distance approach. The shortest distance between two points on a sphere, measured along the Earth surface is known as great circle distance. Alternative to this approach is Law of cosines which is actually accurate only for shorter distance. The equation (9) is used to compute the great circle distance as defined by Haversine approach, where ψ_1 and ψ_2 represent the latitudes of the point 1 and point 2, λ_1 and λ_2 represent the longitude of point 1 and 2 respectively and r is the radius of the Earth at a constant 6371 km.

$$\text{distance} = 2(r) \arcsin \sqrt{\sin^2\left(\frac{\psi_2 - \psi_1}{2}\right) + \cos(\psi_1) \cos(\psi_2) \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}. \tag{9}$$

Table 1: Notations and descriptions.

Notations	Description
S	A set of switches present in the network topology, $S = \{s_1, s_2, s_3, \ldots, s_n\}$, where n is the total number of switches. $l(s)$ is the load associated with the switch
C	A set of controllers to be installed in the network, $C = \{c_1, c_2, \ldots, c_k\}$, where k is the number of controllers to be installed. $L(c)$ is the total capacity associated with the controller
P	The set of all possible locations for controllers' placement
$T(c)$	The set of forwarding switches connected by controller
$d(s,c)$	A function that calculates the shortest path between a switch $s \in S$ to one of the controller $c \in C$
$d(c_i, c_j)$	A function that calculates the shortest path from controller c_i and c_j
x_{cp}	A decision variable that is:= $\begin{cases} 1 & \text{if } c \in C \text{ is installed on location } p \in P \\ 0 & \text{otherwise} \end{cases}$

The mapping of the switch to controller is the relationship of how the switches are controlled by the controller. The assumption is that one switch can be controlled by only one controller and is based on the shortest distance between the switch and the controller. Also, the switch positions where the controllers are placed are controlled by default by the deployed controller. The selection of the second nearest controller is only deemed necessary in situation when the load of the controller exceeds the switch handling capacity. An index list is created which represents the mapping relationship of controllers with the switch as shown in Figure 1.

3.2. Dataset. The dataset used in this study are the real topologies that are obtained from the Internet Topology Zoo (https://www.topology-zoo.org/) and the standard topology e.g. "Savvis" and "Ernet." Savvis is the network topology representing the backbone network of USA, which consists of 19 nodes and 20 edges connection between them. "Ernet" is the backbone network of India which consists of 16 nodes and 18 edges.

3.3. Naked Mole-Rat Controller Placement Problem (NMRCPP). NMR algorithm is used for optimizing the controllers' placement with the motive to minimize controller-controller and controller-switch latency while keeping the load balancing as the constraint. The CPP can be addressed by using NMR algorithm as follows. In NMR algorithm, the solution of the optimization problem is determined by the position of the NMRs. First, number of NMRs is initialized, with each NMR representing one of the combinations of k possible placement of controllers to be deployed. Given the k number of controllers to be deployed, each NMR is represented as a $2k$ dimensional vector and each dimension represents one of the controller position. After initializing the NMRs, each of the controllers' position is assigned the switches in the network based on the shortest distance to the controllers and label for switches mapping is created for every controller. After mapping of the switches to the controller, the load of the controller is measured by using equation (6). If the capacity of controller exceeds its limitation, the switch is assigned to the second nearest controller. As a switch is assigned to the possible nearest controller, the delay from

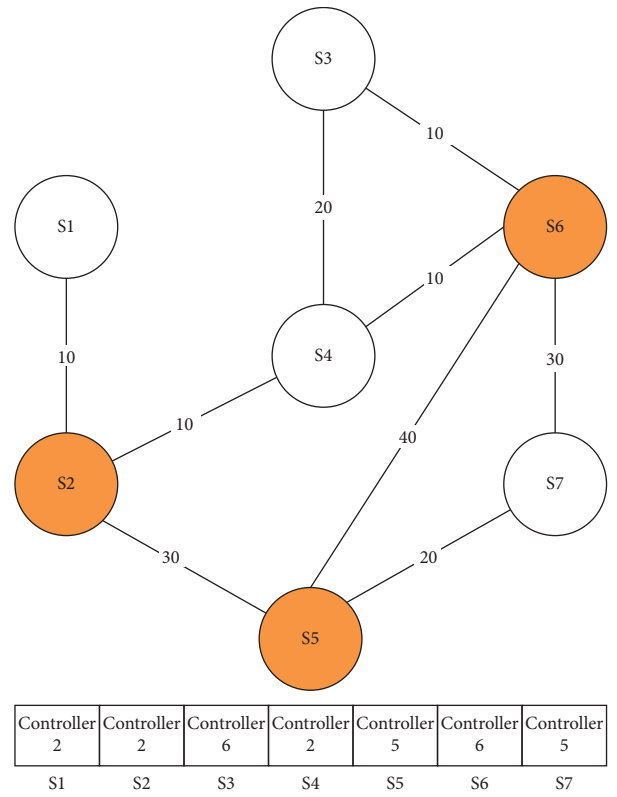

Figure 1: Switch-to-controller mapping.

a node to a nearest controller is measured in terms of distance. Then, the objective function is calculated using equation (8), which represents the fitness of the NMRs. The best b breeders are selected based on the best fitness value and remaining NMRs are selected as workers. Also, the best position of the NMR is set as global best position. Then, until the termination criterion is met, the position of breeders and workers are updated. For all the breeders, the position is updated using equation (3). Breeder NMRs are used for exploitation of search space, thus the search space of breeders is situated around those controllers' position which obtains the best fitness value. Hence, breeder NMRs position is always directed by the best NMR position. After the position is updated, the fitness of the breeders is updated using equation (7) and if it improves the fitness, then the new position is remembered.

Input: LatLong, nCont, SrtPathMtrx /* Latitude & longitude, number of controllers, the shortest distance between each nodes */
Output:best_latency (Optimized Cost), FinalCtrlPos, Mapping$S2C$/*best value of minimized latency, latitude and longitude of the controllers, switch-to-controller mapping relationship */
Initialization:
  nSwitch←size of Latlong //total number of switches
  maxIter←maximum number of iterations
  nPop←NMR swarm size/*total number of NMR population */
  bp←breeding probability/* the breeding probability of the breeder NMR */
  n_B←nPop/5 //population size of the breeder NMR
  for$i$ = 1 to nPop**do**
    $nmr\,(i).Pos$←randomly select nCont from nSwitch
    label←calculate mapping of switch to controller
    fitness←evaluate latency of $nmr\,(i)$
  end for
  Optimization() /*include function optimization*/
Function Optimization() /* To be included in Algorithm 1*/
  whileiter < maxIter**do**
    sort NMRs according to fitness in ascending order
    X_{best}←NMR with best minimum latency
    L_{best}←**label of the best NMR** //mapping of controller and switches
    for$i$ = 1 to n_B**do**
      if$U\,(0, 1) > bp$**then**
        update position using: $b_i^{t+1} = (1-\beta)b_i^t + \beta\,(X_{\text{best}} - b_t^i)$
        //β refers to random (0, 1)
        if latency of $b_i^{t+1} < b_i^t$**then**
          update the new position to b_i^{t+1}
        end if
      end if
    end for
    for$i = n_B + 1$ to nPop**do**
      update position using: $w_i^{t+1} = w_i^t + L \times (w_j^t - w_k^t)$
      //L is the levy flight step
      if latency of $w_i^{t+1} < w_i^t$**then**
        update the new position of: w_i^{t+1}
      end if
      end for
      iter←iter + 1
      Final Ctrl Pos←position of X_{best}
      Mapping$S2C$←label of L_{best}
      best_latency←fitness value of X_{best}
    end while

Algorithm 1: Naked mole-rat controller placement problem.

Similarly, for all the workers, the position is updated using equation (2). The main purpose of workers NMR is exploration of the search space. The exploration of the search space is based on the two random worker NMRs position, thus the update of position vector of worker NMRs is directed by the two randomly selected workers position. Thus, like a random search in the search space, worker NMR performs the exploration part. After updating the position of workers and breeders, their fitness is evaluated and only if the new fitness is better than the previous one, it is stored in the memory. Then, evaluation of the fitness of the NMRs is done and only the best NMRs are selected as the breeder for next iterations. The best among the breeders is memorized as the best solution for that iteration. Finally, when the termination criterion is met, the best breeder NMR presents the solution of the controller placement problem. Thus, the solution contains the position of the controllers, which essentially means the longitude and latitude of the controllers with switch to controllers mapping relationship and the best minimized value of the objective function.

Algorithm 1 elucidates the flow of how the NMRA works for solving the controller placement problem and Algorithm 2 presents the steps of fitness evaluation. The optimization() function is called form Algorithm 1. This function provides the steps to calculate best latency and optimal location. The positional update of worker is based on L, which is the levy flight steps and the two randomly selected worker NMRs while breeders are updated based on β which is the random number between 0 and 1 and the best breeder, which is

```
Input:ShrtPathMtrx, nmr.pos //shortest path matrix containing distance between the nodes, the original controller position
Output:fitness, label/*the average SC latency, mapping of switch-to-controller*/
  for each nmrdo
    for each node do
      find the shortest distance path to the controller;
      label = assign the node nearest to it
      Compute controller load capacity using (6)
      while controller load exceeds the capacity do
        assign switch to the next nearest controller
        update the label
      end while
    end for
    for each controller do
      find the shortest distance path to controller;
      evaluate the controller-controller latency using the path
    end for
    fitness = the combined global average latency using (7)
  end for
  Returnfitness, label
```

Algorithm 2: Evaluating fitness.

basically the best NMR. Here, each NMR represents the solution of the controller placement containing the co-ordinates of the nodes to be deployed.

4. Results and Analysis

The experimental work was carried out in Google Colab. The results are obtained by running the algorithms repeatedly 30 times. The best controller placement is analyzed based on SC latency and SC + CC latency combined. Also, in some scenarios, where load is unbalanced, load balancing is also maintained. The two algorithms NMR and BAT algorithm are analyzed from three aspects: (i) effect on average SC latency with increasing number of the controllers, (ii) effect on global latency with increase in number of controllers (iii) effect of algorithm execution complexity with increase in controllers to find optimal solution.

4.1. Average SC Latency vs. Number of Controllers. Average SC latency in SDWAN refers to the average time taken by the switches to receive its flow table instruction from its nearest associated controllers, which is proportional to the distance between the switch and the controller. The placement of the controller is analyzed based on the following topology:

Case I. SC Latency in Ernet Topology.

In this case, both the NMR and BAT algorithm performed better in terms of the exact same placement for the controllers as shown in Figure 2. The number of controlled switches at different number of nodes for different number of controllers from Figures 2 and 3 are shown in Table 2.

The Figure 4 illustrates the upshots on average SC latency with the increase in the number of the controllers. This can also be represented in Table 3. Thus, it can be concluded that, with the increase in number of the controllers', the average SC latency keeps on decreasing, until there is a controller for a switch. The paramount change in SC latency is seen when the controller number is two when compared to single controller with almost 40% curtailment of latency. The further considerable decrease in SC latency is seen until $k = 4$, as each controller resides in the nodes position with maximum degree available.

Case II: SC Latency in Savvis Topology.

When considering SC latency only, the placement of the controllers is same for $k = 1$, $k = 2$, and $k = 4$, thus both the algorithms have achieved the same SC latency. Figure 5 shows the placement of both NMR and BAT for $k = 2$ and $k = 4$.

In Table 5, the total switches controlled by the three controllers at different places in case of BAT and NMR algorithm is shown. Out of three controllers located, both the BAT algorithm and NMR algorithm have selected Los Angles (node 12) and Saint Louis (node 18) as common nodes to place the controllers, while BAT algorithm selected New York (node 0) as third controller's location, whereas NMR selected Philadelphia (node 1) as controller placement location. The average SC latency of NMR is slightly better than BAT algorithm.

The average SC latency for both scenario of NMR/BAT algorithm for two controllers is presented in Table 6. This table shows unevenly distributed load as well as slightly better load balanced controller by re-assigning the number of switches for the given controllers.

The Figure 6 portrays the SC latency decreasing with increasing number of the controllers. This behavior is clearly mentioned in Table 7. In average SC latency at topology Savvis, NMR algorithm slightly performed better at finding the minimum average latency when compared to BAT algorithm.

Table 2: Controller and switches placement details of Ernet topology.

Number of controllers (K)	Number of controlled switches				
	Node (8), Delhi	Node (7), Bengaluru	Node (13), Kolkata	Node (3), Mumbai/Pune	Node (12), Allahabad
1	Master node	—	—	—	—
2	9 switches, nodes: 4, 5, 8–14	7 switches, nodes: 0–3, 6, 7, 15	—	—	—
3	6 switches, nodes: 4, 5, 8, 10, 11, 12	7 switches, nodes: 0–3, 6, 7, 15	3 switches, nodes: 9, 13, 14	—	—
4	6 switches, nodes: 4, 5, 8, 10, 11, 12	4 switches, nodes: 2, 6, 7, 15	3 switches, nodes: 9, 13, 14	3 switches, nodes: 0, 1, 3	—
5	3 switches, nodes: 4, 5, 8	4 switches, nodes: 2, 6, 7, 15	3 switches, nodes: 9, 13, 14	3 switches, nodes: 0, 1, 3	3 switches, nodes: 10, 11, 12

Table 3: Average SC latency with increasing number of controllers for Ernet topology.

Controller count (K)	Average SC latency (ms)	% decrease in latency from previous
1	6.28	
2	3.75	40.3
3	2.52	32.8
4	1.8	28.6
5	1.53	15.0

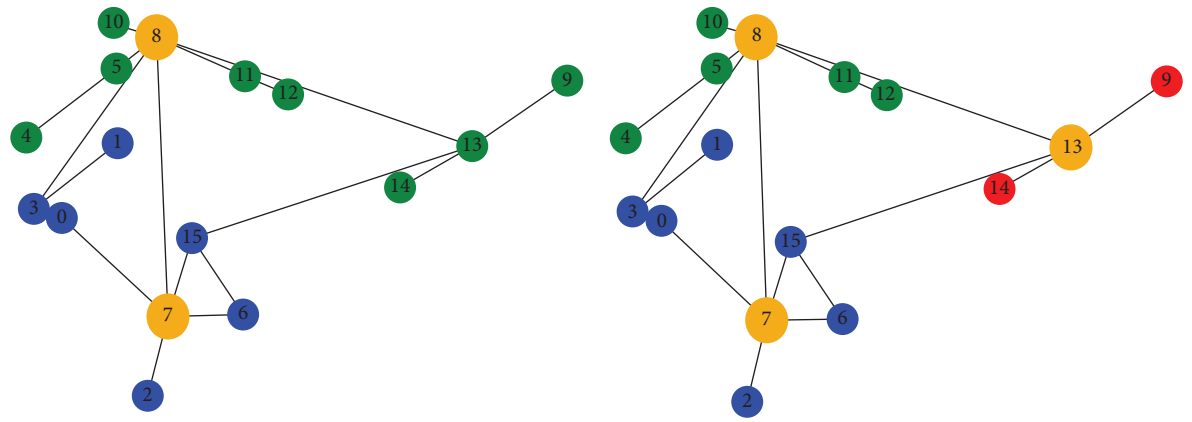

Figure 2: Ernet topology controller placement for (a) $k=2$, (b) $k=3$.

4.2. Average SC and CC Latency vs. Number of Controllers. If the consideration is given to combined SC latency and CC latency, there are two scenarios: (i) 0.9 SC and 0.1 CC (ii) 0.8 SC and 0.2 CC. The analysis is topology specific and mentioned below:

4.2.1. For Ernet Topology with 0.9 SC and 0.1 CC. In this scenario, 90% priority is given to average SC latency and 10% to average CC latency. For two controllers, i.e. $k=2$, and three controllers, i.e. $k=3$, also mentioned in Table 8, both NMR and BAT algorithm has the exact same placement and thus the same latency. However, due to addition of the CC latency, the algorithms focuses on the CC latency as well.

Also, the illustration of both BAT and NMR algorithm as described in Table 9 for the placement of four controllers shows the latency of NMR is better than BAT algorithm.

From the Figure 7, it is observed that latency actually decreases from $k=2$ to $k=3$ scenario, which is due to decrease in SC latency with increase in controller, while addition of a single controller does not increase much CC latency. Further addition of the controllers, i.e. $k=4$ and $k=5$, the CC latency increases significantly due to addition of control path between the controller while there is decrease in SC latency due to addition of the controller, thus overall increasing the global latency with increase in controller. When comparing NMR with BAT, the global latency of NMR is slightly better than BAT algorithm in

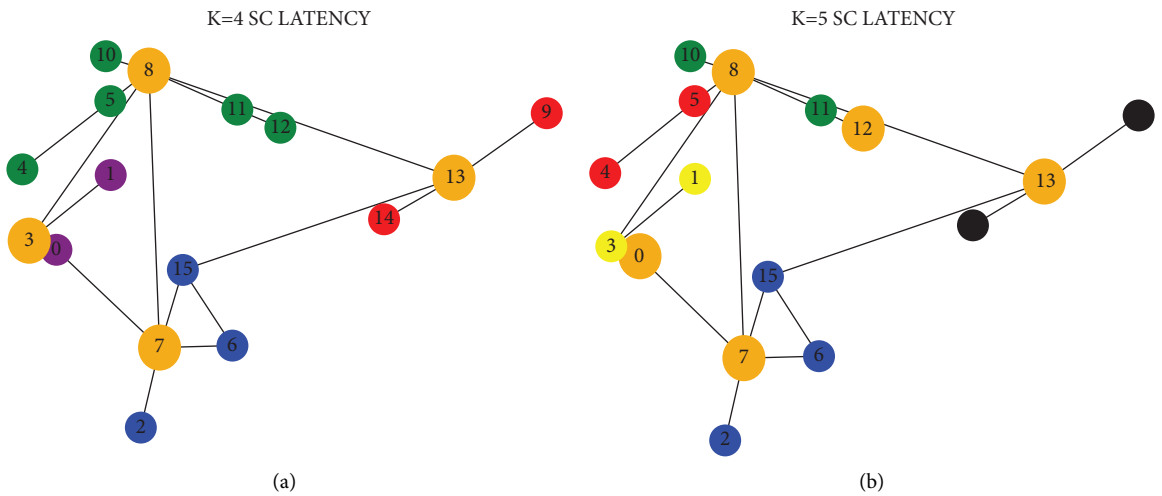

Figure 3: Ernet topology controller placement for (a) $k = 4$, (b) $k = 5$.

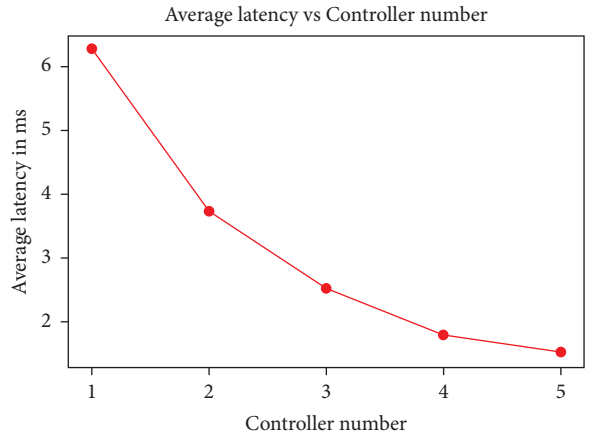

Figure 4: Average SC latency with increasing number of controllers for Ernet topology.

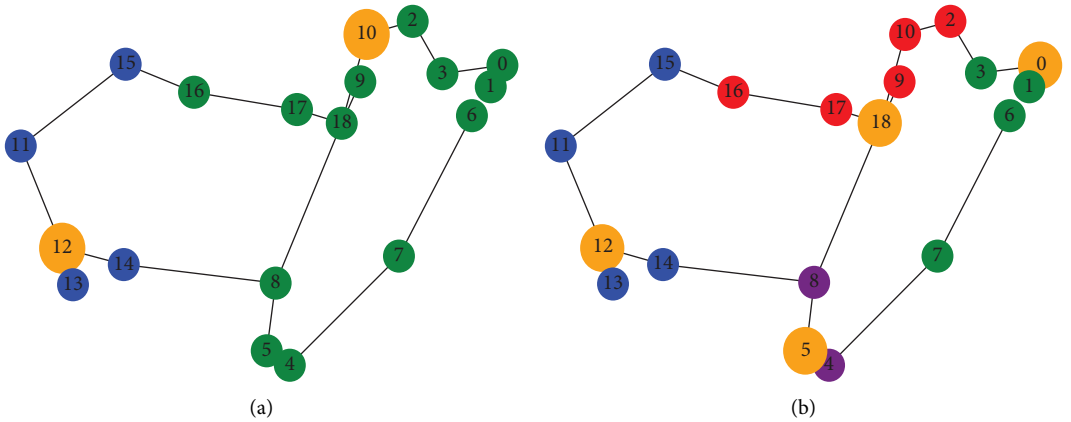

Figure 5: Savvis topology controller placement considering SC latency for (a) $k = 2$, (b) $k = 4$.

Table 4: Controller and switches placement details of Savvis.

Number of controllers (K)	Number of controlled switches				
	Node (10), Chicago	Node (12), Los Angeles	Node (18), Saint Louis	Node (0), New York	Node (5), Austin
2 (BAT/NMR algorithm)	14 switches, nodes: 0–10, 16-18	5 switches, nodes: 11–15	—	—	—
4 (BAT/NMR algorithm)	—	5 switches, nodes: 11–15	6 switches, nodes: 2, 9, 10, 16–18	5 switches, nodes: 0, 1, 3, 6, 7	3 switches, nodes: 4, 5, 8

Table 5: Controller and switch placement details of Savvis for BAT and NMR algorithms.

Number of controllers (K)	Number of controlled switches			
	Node (12), Los Angeles	Node (18), Saint Louis	Node (0), New York	Node (1), Philadelphia
3 (BAT algorithm)	5 switches, nodes: 11–15	9 switches, nodes: 2, 4, 5, 8–10, 16–18	5 switches, nodes: 0, 1, 3, 6, 7	—
3 (NMR algorithm)	5 switches, nodes: 11–15	9 switches, nodes: 2, 4, 5, 8–10, 16–18	—	5 switches, nodes: 0, 1, 3, 6, 7

Table 6: Average SC latency for BAT and NMR algorithms.

Number of controllers (K)	Number of controlled switches			Average SC latency (ms)	Increase in latency from previous
	Node (12), Los Angeles	Node (10), Chicago	Node (0), New York		
2 (NMR/BAT) algorithm	5 switches, nodes: 11–15	14 switches, nodes: 0–10, 16–18	—	5.08	—
2 (NMR/BAT) algorithm	8 switches, nodes: 5, 8, 11–16	—	11 switches, nodes: 0-4, 6–10, 17, 18	5.61	10%

scenario when 90% priority is given to SC latency, while 10% priority is given to CC latency.

4.2.2. For Ernet Topology with 0.8 SC and 0.2 CC. In this scenario, 80% priority is given to SC latency and 20% priority is given to CC latency. For $k = 2$–5 both NMR and BAT algorithm have the same placement of controller and thus have same global average latency which is summarized in Table 10.

Due to the same placement of the controllers in this scenario, both NMR and BAT algorithms have same global average latency as shown in Figure 8. When the controller is increased from $k = 2$ to $k = 3$, the global average latency has increased by 8%, while for scenario when additional controller is added, i.e. $k = 4$, the average latency increases by almost 14% and when there are 5 controllers, there is increase in latency by almost 19% compared to 4 controllers scenario. The observed increase in latency is due to additional path cost due to increase in number of controllers. Whenever, a new controller is added, the shortest path from all the previous controllers needs to be assessed and calculated, which ultimately contributes to increase in global average latency.

4.2.3. Load Balancing. In most of the scenarios, both NMR and BAT algorithm provided the controller placement with almost balanced load condition. Here, a balanced load is considered, when there is minimum difference between maximum numbers of switches controlled by the controllers in a certain placement of controllers. However, there are some specific scenarios where algorithms completely disregard the load balancing and just focus on minimizing the latency, thus creating unbalanced load scenarios.

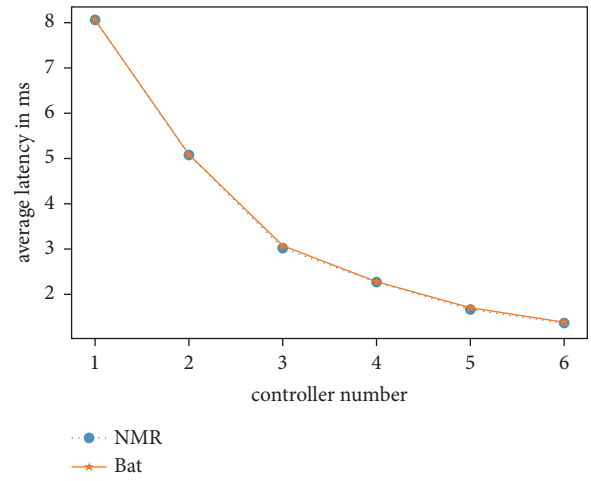

Figure 6: Average SC latency with increasing number of controllers for Savvis topology.

Table 7: Average SC latency with increasing number of controllers for Savvis topology.

Number of controllers (K)	Average SC latency (ms) NMR algorithm	Average SC latency (ms) BAT algorithm	% decrease in latency from previous
1	8.06	8.06	
2	5.08	5.08	37.0
3	3.01	3.05	40.8
4	2.27	2.27	24.6
5	1.66	1.69	26.9
6	1.38	1.38	16.9

Table 8: Ernet: controller placement for $K = 2$ and 3 using both NMR and BAT at 0.9 SC + 0.1 CC scenario.

Number of controllers (K) at 0.9 SC + 0.1 CC	Number of controlled switches			
	Node (8), Delhi	Node (0), Pune	Node (3), Mumbai	Node (7), Bengaluru
2 (NMR/BAT) algorithm	9 switches, nodes: 4, 5, 8–14	7 switches, nodes: 0–3, 6, 7, 15 switches	—	—
3 (NMR/BAT) algorithm	9 switches, nodes: 4, 5, 8–14	—	3 switches, nodes: 0, 1, 3	4 switches, nodes: 2, 6, 7, 15

Table 9: Ernet: controller placement for $K = 4$ using both NMR and BAT at 0.9 SC + 0.1 CC scenario.

Number of controllers (K) at 0.9 SC + 0.1 CC	Number of controlled switches				
	Node (8), Delhi	Node (3), Mumbai	Node (13), Kolkata	Node (7), Bengaluru	Node (6), Chennai
4 (BAT algorithm)	6 switches, nodes: 4, 5, 8, 10–12	3 switches, nodes: 0, 1, 3	3 switches, nodes: 9, 13, 14	4 switches, nodes: 2, 6, 7, 15	—
4 (NMR algorithm)	6 switches, nodes: 4, 5, 8, 10–12	3 switches, nodes: 0, 1, 3	3 switches, nodes: 9, 13, 14	—	4 switches, nodes: 2, 6, 7, 15

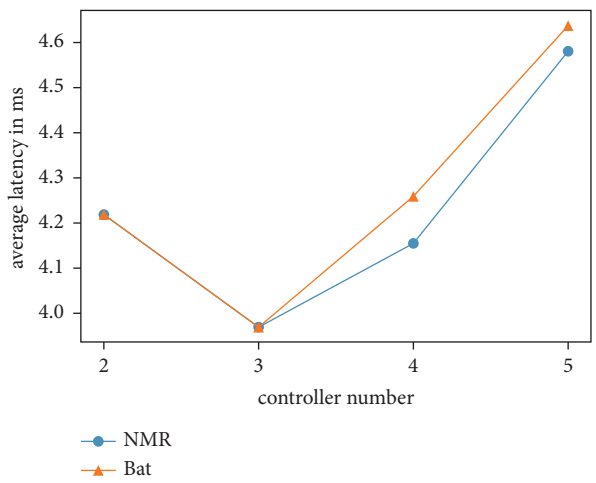

Figure 7: Average latency with increasing controllers at 0.9 SC + 0.1 CC in Ernet topology.

Table 10: Ernet: controller placement for $K = 2$ and 3 using both NMR and BAT at 0.8 SC + 0.2 CC scenario.

Number of controllers (K) at 0.8 SC + 0.2 CC	Number of controlled switches		
	Node (8), Delhi	Node (3), Mumbai	Node (7), Bengaluru
2 (NMR/BAT) algorithm	9 switches, nodes: 4, 5, 8–14	7 switches, nodes: 0–3, 6, 7, 15	—
3 (NMR/BAT) algorithm	9 switches, nodes: 4, 5, 8–14	3 switches, nodes: 0, 1, 3	4 switches, nodes: 2, 6, 7, 15

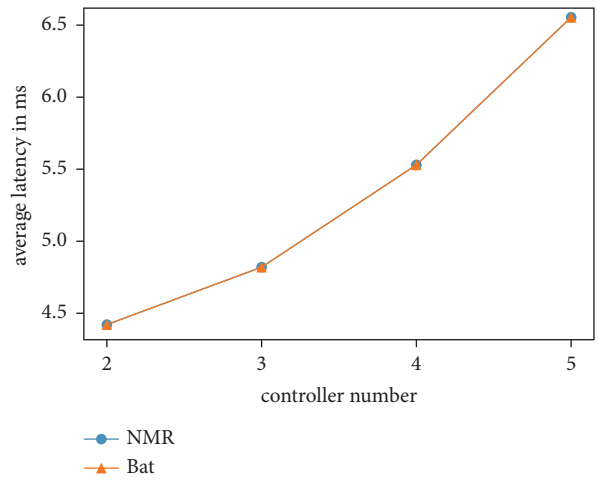

Figure 8: Change in global average latency with increase in number of controllers at 0.8 SC + 0.2 CC scenario in Ernet topology.

Table 11: Ernet: controller placement for $k = 3$ at load unbalanced/balanced (0.9 SC + 0.1 CC) scenario.

Number of controllers (K)	Number of controlled switches				Average SC latency (ms)	Latency increase from previous
	Node (8), Delhi	Node (3), Mumbai	Node (7), Bengaluru	Node (13), Kolkata		
3 (NMR/BAT) algorithm	9 switches, nodes: 4, 5, 8–14	3 switches, nodes: 0, 1, 3	4 switches, nodes: 2, 6, 7, 15	—	3.97	—
3 (NMR/BAT) algorithm	6 switches, nodes: 4, 5, 8, 10–12	7 switches, nodes: 0–3, 6, 7, 15	—	3 switches, nodes: 9, 13, 14	4.26	5%

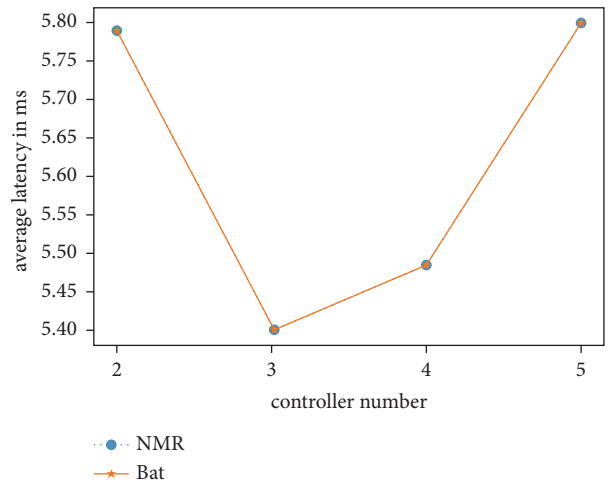

Figure 9: Comparison of global average latency variation at 0.9 SC + 0.1 CC in Ernet topology.

Table 11 presents the load balancing scenario and respective global average latency output. Figure 9 shows the global average latency output variation from the previous controller's size while increasing number of controllers from 2 to 5. As shown in Table 12, while $K = 3$, the global average latency reduces from 5.79 ms to 5.4 ms. Even though the addition of another controller reduces the SC, the further addition of controller i.e., $k = 4$ and 5 causes increase in

Table 12: Global average latency variation at 0.9 SC + 0.1 CC for NMR and BAT algorithms.

Number of controllers (K) at 0.9 SC + 0.1 CC	Global average latency (ms)
2	5.79
3	5.4
4	5.49
5	5.81

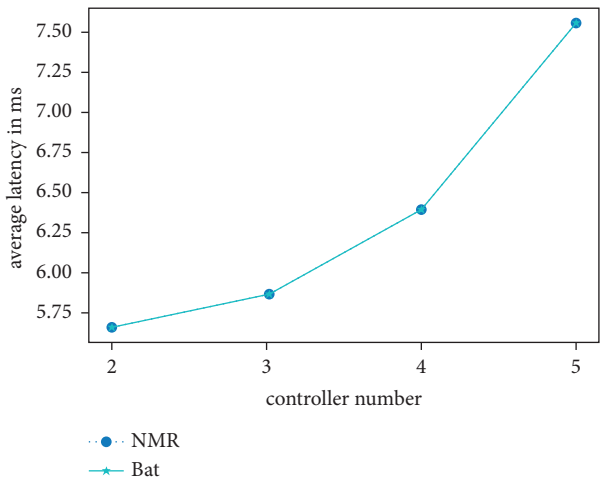

Figure 10: Global average latency variation at 0.8 SC + 0.2 CC in Ernet topology.

Table 13: Comparison of global average latency variation at 0.8 SC + 0.2 CC in Ernet topology.

Number of controllers (K)	Global average latency (ms)	Latency increase from previous
2	5.66	—
3	5.86	3.53%
4	6.4	9.22%
5	7.55	17.97%

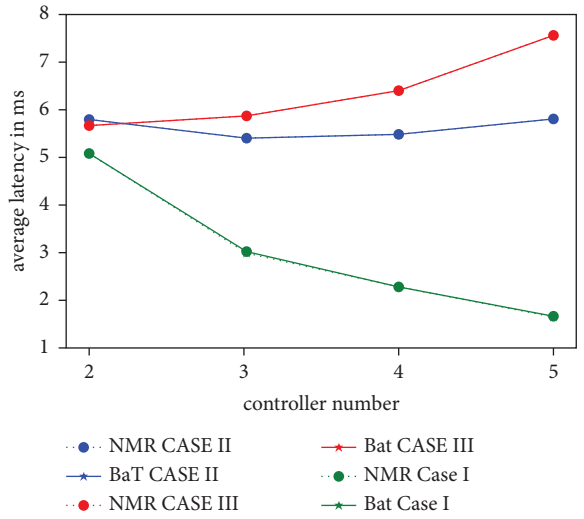

Figure 11: Global average latency variation for different cases of NMR and BAT algorithms.

Table 14: Global average latency variation with different cases for SC and CC using NMR and BAT algorithms.

Cases	Average SC latency	Global average latency	Remarks
I. Only SC	Decreasing with increasing controller number	—	—
II. 0.9 SC + 0.1 CC	—	Not decreasing with increasing controller number	Slight decline at $k = 2$ and 3 and slight increment after after $k = 4$
III. 0.8 SC + 0.2 CC	—	Starts rising with increasing controller number	Effect of increase in CC latency

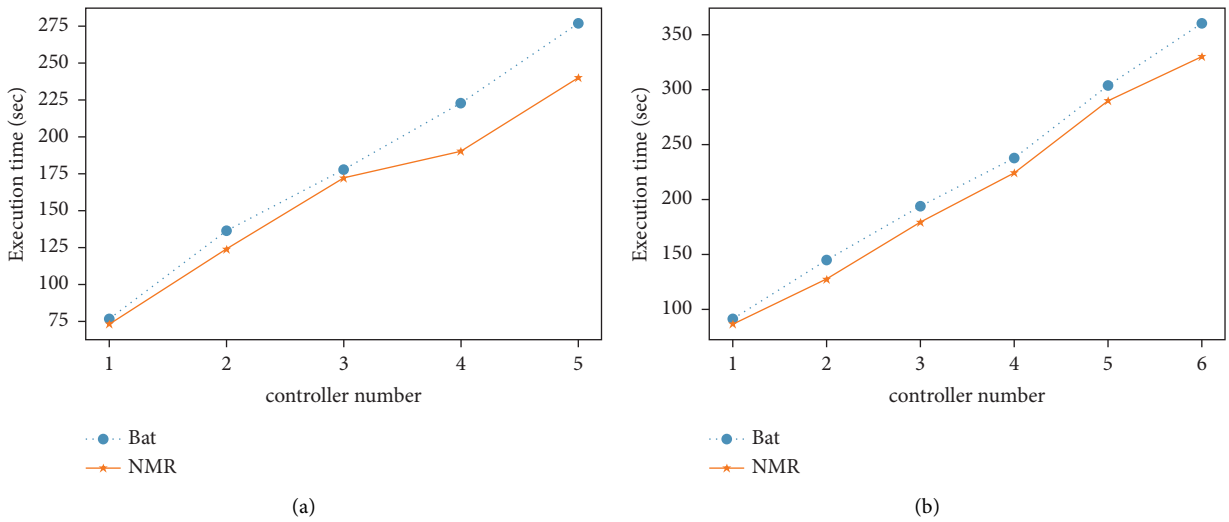

Figure 12: Execution time of algorithms with increase in number of controller in (a) Ernet, and (b) Savvis toplogies.

control paths between the controllers and hence, the result is in increasing CC.

For the scenario of Ernet topology at 0.8 SC + 0.2 CC, the placement of controllers for all the scenarios ($k = 2$ to 5) both NMR and BAT algorithm is exactly same, thus the algorithms give the same global average latency. Even though there is increase in controller number, the global average latency has not increased much because with increase in controller the average SC latency decreases, which contributes in 80% of global latency whereas the average CC latency increase, thus ultimately increasing the global latency at three controller scenario.

As shown in Figure 10 and summarized in Table 13, as the number of controllers increases, the global average latency also increases. With the increase in number of the controllers, the best control paths needs to be located with all the controllers, thus even a 20% contribution on global latency adds a significant part of average CC latency to average global latency. A comparison of different case scenarios has been presented in Figure 11 and summarized in Table 14 for different controllers.

4.3. Execution Complexity. Execution complexity refers to time taken by algorithms to find the best placement of the controller. The Figure 12 illustrates that with the increase in number of controllers, the execution time increases, as with the increase in number of controllers, the placement matrix in the algorithm increases. Due to this increase in placement matrix, the possible combinations of the placement increases which results in complexity of algorithms. Figure 12 shows that the NMR algorithm slightly performed better. When comparing two topologies, the algorithm complexity increases with increase in number of nodes as the execution time of Savvis topology is higher than that of Ernet topology.

5. Conclusion and Future Work

The analysis of the optimum controller placement considered in this study takes into account both the SC latency and global latency (SC + CC latency). Three aspects: (i) effect on average SC latency (ii) effect on global latency (iii) effect of algorithm execution complexity with increase in controllers are included in the analysis of the two algorithms NMR and Bat to find the optimal solution. The ideas and mechanisms are illustrated using two publicly available standard topologies viz. Ernet and Savvis. The controller localization approach implemented with NMR algorithm has slightly a better result as compared with the Bat algorithm.

This research was limited to finding the controller placement considering average SC latency, average CC latency and load balancing. Load balancing is based on the assumption that every controller has a fixed capacity up to which the processing latency can be considered negligible. Thus, if the difference between the maximum and minimum number of switches connected can be reduced, a slightly load balanced placement can be achieved, as it reduces the maximum controller utilization for each controller. The work can be extended further by considering the processing latency of the controller. Further works can be done to bolster the reliability, fault tolerance and minimize the cost of placement. Additionally, this study is limited to present NMR algorithm as an approach for multi-controller placement over SDN environment and made a comparison with Bat algorithm. The comparison with other state-of-the-art approaches is considered as future recommendations, for example, the BFR approach can be implemented for multi-controller placement optimization.

Data Availability

The manuscript contains all the data used in the study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by University Grants Commission (UGC), Nepal (Grants ID: CRG-078/79-Engg-01).

References

[1] B. R. Dawadi, D. B. Rawat, S. R. Joshi, P. Manzoni, and M. M. Keitsch, "Migration cost optimization for service provider legacy network migration to software-defined ipv6 network," *International Journal of Network Management*, vol. 31, no. 4, Article ID e2145, 2020.

[2] A. Kumari and A. S. Sairam, "Controller placement problem in software-defined networking: a survey," *Networks*, vol. 78, no. 2, pp. 195–223, 2021.

[3] Y. Fan, L. Wang, and X. Yuan, "Controller placements for latency minimization of both primary and backup paths in sdns," *Computer Communications*, vol. 163, pp. 35–50, 2020.

[4] B. R. Dawadi, D. B. Rawat, S. R. Joshi, and P. Manzoni, "Legacy network integration with sdn-ip implementation towards a multi-domain sodip6 network environment," *Electronics*, vol. 9, no. 9, p. 1454, 2020.

[5] G. Wang, Y. Zhao, J. Huang, and Y. Wu, "An effective approach to controller placement in software defined wide area networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 344–355, 2018.

[6] B. Isong, R. R. S. Molose, A. M. Abu-Mahfouz, and N. Dladlu, "Comprehensive review of sdn controller placement strategies," *IEEE Access*, vol. 8, Article ID 170070, 2020.

[7] A. K. Singh and S. Srivastava, "A survey and classification of controller placement problem in sdn," *International Journal of Network Management*, vol. 28, no. 3, Article ID e2018, 2018.

[8] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *ACM SIGCOMM - Computer Communication Review*, vol. 42, no. 4, pp. 473–478, 2012.

[9] N. Firouz, M. Masdari, A. B. Sangar, and K. Majidzadeh, "A novel controller placement algorithm based on network portioning concept and a hybrid discrete optimization algorithm for multi-controller software-defined networks," *Cluster Computing*, vol. 24, no. 3, pp. 2511–2544, 2021.

[10] R. Salgotra and U. Singh, "The naked mole-rat algorithm," *Neural Computing & Applications*, vol. 31, no. 12, pp. 8837–8857, 2019.

[11] H. Huang, H. Yin, G. Min, H. Jiang, J. Zhang, and Y. Wu, "Data-driven information plane in software-defined networking," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 218–224, 2017.

[12] Si-K. Yoon, Zia Khalib, N. Yaakob, and A. Amir, "Controller placement algorithms in software defined network-a review of trends and challenges," in *Proceedings of the MATEC Web of Conferences*, vol. 140, EDP Sciences, Article ID 01014, 2017.

[13] T. Hu, Z. Guo, P. Yi, T. Baker, and J. Lan, "Multi-controller based software-defined networking: a survey," *IEEE Access*, vol. 6, pp. 15980–15996, 2018.

[14] H. Babbar, S. Rani, and M. Masud, "Load balancing algorithm for migrating switches in software-defined vehicular networks," *Computers, Materials & Continua*, vol. 67, 2021.

[15] R. Salgotra, U. Singh, G. Singh, N. Mittal, and A. H. Gandomi, "A self-adaptive hybridized differential evolution naked mole-rat algorithm for engineering optimization problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 383, Article ID 113916, 2021.

[16] K. A. Rasol and J. Domingo-Pascual, "Evaluation of joint controller placement for latency and reliability-aware control plane," in *Proceedings of the 8th International Conference on Software Defined Systems (SDS)*, pp. 1–7, IEEE, Gandia, Spain, December 2021.

[17] G. Wang, Y. Zhao, J. Huang, and W. Wang, "The controller placement problem in software defined networking: a survey," *IEEE Network*, vol. 31, no. 5, pp. 21–27, 2017.

[18] J. Cui, Q. Lu, H. Zhong, M. Tian, and Lu Liu, "A load-balancing mechanism for distributed sdn control plane using response time," *IEEE transactions on network and service management*, vol. 15, no. 4, pp. 1197–1206, 2018.

[19] E. Calle, D. Martínez, M. Mycek, and M. Pióro, "Resilient backup controller placement in distributed sdn under critical targeted attacks," *International Journal of Critical Infrastructure Protection*, vol. 33, Article ID 100422, 2021.

[20] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A survey of controller placement problem in software-defined networking," *IEEE Access*, vol. 7, Article ID 24290, 2019.

[21] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, and T. Li, "Density cluster based approach for controller placement problem in large-scale software defined networkings," *Computer Networks*, vol. 112, pp. 24–35, 2017.

[22] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "Efficient controller placement and reelection mechanism in distributed control system for software defined wireless sensor networks," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 6, Article ID e3588, 2019.

[23] A. K. Singh, S. Maurya, N. Kumar, and S. Srivastava, "Heuristic approaches for the reliable sdn controller placement problem," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, Article ID e3761, 2020.

[24] A. K. Singh, S. Maurya, and S. Srivastava, "Varna-based optimization: a novel method for capacitated controller placement problem in sdn," *Frontiers of Computer Science*, vol. 14, no. 3, Article ID 143402, 2020.

[25] L. Liao, V. C. M. Leung, Z. Li, and H.-C. Chao, "Genetic algorithms with variant particle swarm optimization based mutation for generic controller placement in software-defined networks," *Symmetry*, vol. 13, no. 7, p. 1133, 2021.

[26] N. S. Radam, S. T. F. Al-Janabi, and K. S. Jasim, "Multi-controllers placement optimization in sdn by the hybrid hsa-pso algorithm," *Computers*, vol. 11, no. 7, p. 111, 2022.

[27] M. Khorramizadeh and V. Ahmadi, "Capacity and load-aware software-defined network controller placement in heterogeneous environments," *Computer Communications*, vol. 129, pp. 226–247, 2018.

[28] S. Guan, J. Li, Yi Li, and Z. Wang, "A multi-controller placement method for software defined network based on improved firefly algorithm," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 7, Article ID e4482, 2022.

[29] A. A. Z. Ibrahim, F. Hashim, N. K. Noordin, A. Sali, K. Navaie, and S. M. E. Fadul, "Heuristic resource allocation algorithm for controller placement in multi-control 5g based on sdn/nfv architecture," *IEEE Access*, vol. 9, pp. 2602–2617, 2021.

[30] C. Gao, H. Wang, F. Zhu, L. Zhai, and S. Yi, "A particle swarm optimization algorithm for controller placement problem in software defined network," in *Proceedings of the International conference on algorithms and architectures for parallel processing*, pp. 44–54, Springer, Zhangjiajie, China, November 2015.

[31] Yi Li, W. Sun, and S. Guan, "A firefly inspired controller placement algorithm in software defined network," in *Proceedings of the IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET)*, pp. 254–258, IEEE, Beijing, China, August 2019.

[32] M. Dhar, B. K. Bhattacharyya, M. Kanti Debbarma, and S. Debbarma, "A new optimization technique to solve the latency aware controller placement problem in software defined networks," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 10, Article ID e4316, 2021.

[33] J. Zhao, H. Qu, J. Zhao, Z. Luan, and Ya Guo, "Towards controller placement problem for software-defined network using affinity propagation," *Electronics Letters*, vol. 53, no. 14, pp. 928-929, 2017.