Computer Standards & Interfaces 97 (2026) 104109

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

DEA-GAO: A two-stage approach optimal controller placement in software-defined networks using data envelopment analysis

Fatemeh Dashti, Ali Ghaffari *, Nahideh Derakhshanfard, Shiva Taghipoureivazi

Department of computer engineering, Ta.C., Islamic azad university, Tabriz, Iran

ARTICLE INFO

Keywords:
Software-defined networking (SDN)
Controller placement problem
Delay
load balancing
reliability

ABSTRACT

Software-defined networks (SDN) have demonstrated considerable benefits in various practical domains by decoupling the control plane from the data plane, thus facilitating programmable network management. This paper presents a two-stage approach for solving the problem of controller placement called DEA-GAO. In the first stage, this strategy assumes the SDN network as a graph and using Data Envelopment Analysis (DEA) and relying on graph centrality metrics such as closeness centrality, betweenness centrality, and eigenvector centrality, calculates the efficiency of nodes to determine the optimal locations for deploying controllers. In the second stage, to allocate switches to controllers, the proposed strategy employs the Green Anaconda Optimization algorithm (GAO) to achieve an optimal allocation while considering network parameters such as average delay, load balancing, and reliability. Finally, to assess the efficacy of the proposed methodology, it is juxtaposed with three extant methods utilizing diverse datasets from the Internet Topology Zoo. The experimental findings indicate that the proposed approach significantly surpasses the existing methods, specifically the hybrid RDMCP-PSO algorithm, heuristic CPP algorithm and PSO algorithm in terms of both average delay (8.8 %, 28.8 % and 22.2 % respectively) and controller utilization (1.5 %, 7.3 % and 32 % respectively).

1. Introduction

The Software-Defined Networking (SDN) paradigm is a sophisticated model supporting dynamism, automation, flexibility, and centralized control of the underlying network that distinguishes it from other conventional networking models. The new paradigm has intrinsic merits like enhanced performance and reduced latency [1]. In SDN, there is a clear distinction between the control plane, which decides on routing, and the forwarding plane, which is responsible solely for packet forwarding based on routing instructions it receives from the control plane (Fig. 1). The planes communicate with each other via Application Programming Interfaces (APIs), e.g., Representational State Transfer (REST) on the northbound interface and OpenFlow on the southbound interface [2,3]. While separating control and data planes offers technical benefits, it also involves several challenges, including how many controllers are required, optimally where they must be placed, and how data plane devices are to be assigned to controllers [1,3].

With the increasing scale of Software-Defined Networking (SDN) infrastructures, the implementation of one controller to handle the entire network has presented more challenges to the SDN controller's processing capacity, more so concerning scalability, performance, and reliability. To mitigate these challenges, the use of multiple controllers is a feasible solution. The multi-controller approach splits the network into smaller domains with every one of them being handled by a specific controller. The set of controllers controlling these regions communicates with one another through their west/eastbound interfaces [1].

For SDN multi-controller networks, the number of deployed controllers is based on the network size and the amount of traffic [2,4]. The major challenges with SDN multi-controller structures involve the number of deployed controllers and the placement of the deployed controllers on distributed switches, typically the controller placement problem (Fig. 2). This problem aims to determine the most suitable locations for SDN controllers to achieve different goals like the reduction of latency, efficiency in terms of power, load balancing, and enhanced reliability [2,4]. The CAPEX and OPEX of the network are greatly influenced by the number of installed controllers. Hence, the best number of controllers and the determination of their most optimal locations are also of prime concern in SDN multi-controller networks [4].

The ideal placement of these controllers directly affects different performance parameters such as latency, reliability, and fault tolerance

* Corresponding author.
E-mail address: A.ghaffari@iaut.ac.ir (A. Ghaffari).

https://doi.org/10.1016/j.csi.2025.104109
Received 12 August 2025; Received in revised form 13 September 2025; Accepted 28 November 2025
Available online 2 December 2025
0920-5489/© 2025 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

which are extremely crucial to ensure a robust network infrastructure. In real-world environments, this problem is further exacerbated by the need to consider possible controller failures that can bring network operations to a standstill [5,6]. The resilient and functional SDN topology should not only be designed to perform optimally under typical operating conditions but also to handle effectively the scenario of one or more controller failures while minimizing interruption and maintaining continuity of services [7]. This need arises from an overall objective of creating Software-Defined Networks that are not only operationally effective and responsive but also inherently resilient and fault-tolerant to the extent of surviving all types of failures without noticeable degradation of service [7,8]. The need to address this issue stems from the fact that the situation in SDN research today calls for adult and highly sophisticated answers to the knotty issue of controller placement, particularly against the backdrop of the dynamic and generally unpredictable nature of network traffic and the inherent risks of component failures [7]. Numerous solutions have been developed to address this multi-dimensional problem based on different network parameters and as per different strategies, such as latency-aware models, traffic load balancing strategies, cost-effective schemes, and overhead-aware control plane solutions [9]. Network reliability-based solutions, quality of service (QoS) concerns, and multi-objective optimization methodologies are also present that attempt to balance a set of conflicting parameters in controller placement [9]. Some researchers employed clustering-based approaches, partitioning the entire network into multiple small clusters, each managed by independent controllers, and ultimately needing synchronization among all of the controllers to offer centralized management of the entire network infrastructure [8].

In this paper, we initially employ Data Envelopment Analysis (DEA) to ascertain the positioning of controllers across all nodes within the network, each of which possesses the potential capability to be designated as a controller. Indeed, the application of the DEA methodology in addressing the controller placement dilemma constitutes a robust and empirical strategy for navigating the intricacies associated with this matter, and through the utilization of this approach, the optimal locations for the controllers can be discerned while accommodating various and often conflicting criteria. The DEA methodology assesses the efficiency of each unit within the Decision-Making Unit (DMU) framework by delineating inputs and outputs corresponding to each unit, thus enabling the prioritization of units based on their efficiency, which facilitates the selection of several high-efficiency units [10]. To establish the input and output parameters, we conceptualized the Software-Defined Networking (SDN) architecture as a graph and employed graph centrality metrics, including degree centrality, closeness, betweenness, eigenvector, dispersion, information, and potential influence as parameters, thereby defining each graph node as a DMU within the DEA framework. The outcome of this methodology delineates the locations of the network controllers. Subsequently, in the second stage of the proposed methodology, we endeavored to optimally allocate the network switches to the identified controllers by utilizing the Green Anaconda optimization algorithm, thus providing an optimal resolution for the controller placement conundrum in SDN networks, which we implemented in Python and simulated within the Mininet environment.

The main contributions of our research are delineated as follows:

- Considering the Software Defined Networking (SDN) infrastructure as a comprehensive graph and meticulously calculating the various centrality metrics that correspond to each node present within the network topology.
- Defining each distinct node within this intricate network as a Decision-Making Unit (DMU) and subsequently calculating the operational efficiency of each DMU to select the most optimal location for the deployment of a controller by employing the Data Envelopment Analysis (DEA) method.
- Formulating the challenge of assigning network switches to their respective controllers is a sophisticated optimization problem that necessitates a thorough analysis and strategic approach to find the most effective solution.
- Utilizing the Green Anaconda Optimization (GAO) methodology to achieve the optimal allocation of network switches to their corresponding controllers, whereby each allocation scenario is conceptualized as a population of anacondas and progresses through various evolutionary stages to ultimately arrive at the most satisfactory solution to the stated optimization problem.

The subsequent sections of this manuscript are organized as follows. Section 2 reviews pertinent literature that has employed evolutionary algorithms in addressing the controller placement issue. Section 3 provides a comprehensive exposition of the proposed DEA-GAO methodology, incorporating the foundational principles of Data Envelopment Analysis (DEA) and the GAO algorithm. Section 4 assesses the effectiveness of the proposed approach. Finally, we synthesize our findings and conclusions in Section 5.

2. Related works

All past research has considered the challenges of the distributed

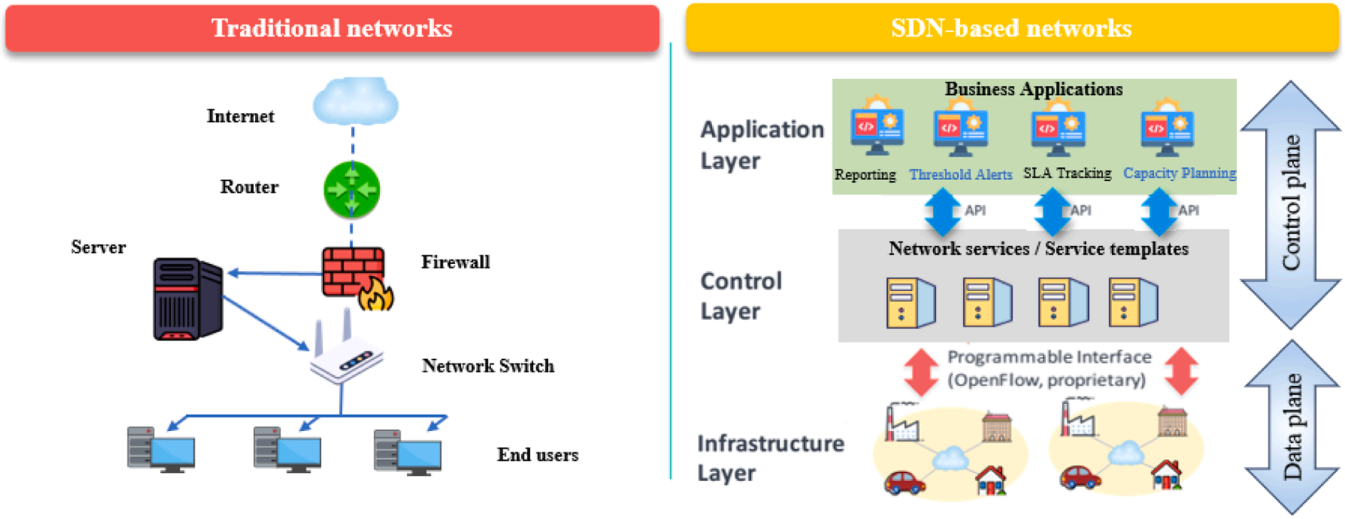


Fig. 1. SDN architecture vs. traditional network architecture.

controller's strategy in two main parts such as: (i) identifying the optimal number of controllers necessary for a specified network topology such that each controller is neither overburdened nor underutilized, and (ii) optimizing the placement of controllers, as this has far-reaching implications for various factors such as flow setup latency, resilience, and load distribution [11,12].

Because this paper employs a biologically inspired evolutionary algorithm known as the Green Anaconda Optimization Algorithm (GAO), this section will delve exclusively into the evolutionary-based methods to optimize the quantity, positioning, and distribution of controllers within the Software-Defined Networking (SDN) milieu. These methodologies are categorized as bio-inspired techniques and Genetic Algorithms (Table 1).

2.1. BIO-inspired CPP

Numerous educational computing methodologies for SDN-oriented Controller Placement Problems (CPP) emphasize the development of hybrid optimization algorithms. Notable algorithms in this domain include Particle Swarm Optimization (PSO) [12], Ant Colony (ACO) [13], and others, summarized in the first part of Table 1.

Thalapala et al. introduced a novel method called Whale Optimization Controller Placement (WOCP) to carefully place the minimum number of controllers in an SDN architecture [11]. The WOCP is inspired by the mathematical model resembling the spiral bubble-net feeding behavior of humpback whales to conduct optimization. By taking advantage of an objective function, this method aims to minimize the average latency of the network via the optimal placement of controllers. They reduce the average network delay by 18 % in MLCP compared to the traditional method, MLCP, by reducing the latency between the switch node and the controller.

Singh et al [12] focused on communication latency and reliable communication as the dual dimensions of the Controller Placement Problem (CPP). The primary object is to suppress the average latency over a network. The test field will be planted with robust network models that can still perform (although less effectively) when n-1 out of n controllers are nonoperational. The form this work is about to take can be seen at its start. By use of an efficient algorithm called "Capacitated Controllers Arrangement (CCA)" to abstract the problem participants system; a nature-inspired metaheuristic technique Particle Swarm Optimization (PSO) is later employed for its resolution. With strategic placement of controllers, this adaptive method dramatically reduces network latency over what would have occurred using the previous method and builds upon switch assignment techniques. By so doing, we believe that not only will this innovative approach better the operation of SDN (Software-Defined Networking), but also it sets standards for later controller deployment methodologies of tomorrow.

Chunlin Li et al. have come up with a dependable controller placement framework in the form of an optimization-based delay and load consideration [13]. Their Load Optimization Controller Placement (LOCP) algorithm marginally performs better than K-means, Simulated Annealing (SA), and Equal-Cost Multi-Path (ECMP) algorithms in propagation delay, queuing delay, and the efficiency of load balancing in massive networks, achieving an average of 18.36 % improvement in load balancing along with optimizing propagation and queuing delays simultaneously. In addition, they proposed a resource allocation algorithm with task delay and reliability parameters along with a resource allocation strategy employing a heuristic ant colony algorithm. This solution addresses the problem of intelligently allocating resources of Mobile Edge Computing (MEC) servers when the server cannot provide task reliability in one channel to minimize average task delay. The experimental results establish that, relative to existing procedures, the resource allocation method employing the Heuristic Ant Colony Algorithm (HACA) effectively optimizes the workload distribution issue and substantially reduces user resource response time and the mean completion delay at the cost of reliability parameters.

In their latest contribution to this area, Tahmasebi et al. have formulated the placement problem of a controller as an optimization problem with reliability, fault tolerance, and performance metrics against latency, synchronization overhead, and deployment cost concerns [14]. They employed a nature-inspired, population-based meta-heuristic algorithm known as SYNCOP for solving the optimization problem with acceptable results.

Abeer A.Z. et al. employed a state-of-the-art heuristic method of current standards to solve the Controller Placement Problem (CPP) with the view to optimize switch distribution and reconfiguration to near controllers in addition to their originally assigned controllers [15]. Their algorithm computes optimal controller placement, minimum required

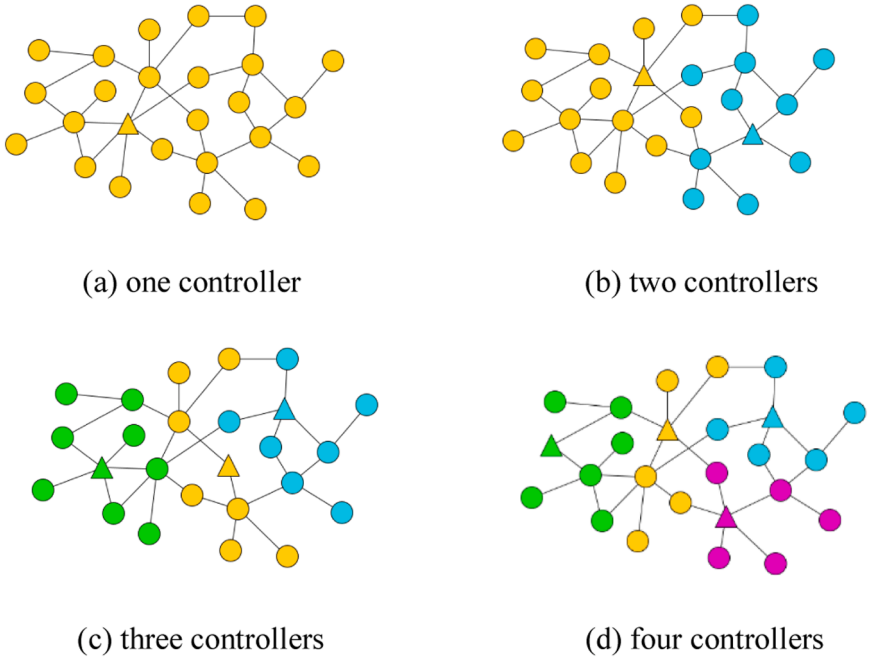


Fig. 2. Controller placement problem with different number of controllers.

Table 1
Evolutionary-based controller placement methods.

Evolutionary Type	Ref.	Method	Implementation	Evaluation metrics	Year
Bio-Inspired CPP	[11]	WOCP: Whale optimization	Simulation: Python programming Hardware: Intel i3 11th generation processor Topology: Nsfnet, Agis, Geant	Latency Reliability	2025
	[12]	PSO	Simulation: MATLAB. Hardware: Intel 1–7 processor,16 GB RAM, 1.6 series Graphic card Topology: Internet topology zoo	Latency	2024
	[13]	Ant colony	Topology: Internet2 OSE3, Janetbackbon	load balancing delay Reliability	2022
	[14]	Cuckoo	Simulation: Java language Hardware: Mac OS, Core i5 2.9 GHz, and 8 GB memory Topology: No standard topology	Reliability fault tolerance latency synchronization overhead deployment cost.	2021
	[15]	RDMCP-PSO	Simulation: MATLAB. Hardware: Intel Core i7-G10 processor and 16 GB RAM Topology: Usnet, Iris-net, Chinanet, Bellcanada, Agis, Aarnet	Reliability load balancing delay	2024
	[16]	APSO	simulation: MATLAB. Hardware: Intel ® core ™ i5–700 U CPU at 2.50-GHz	Delay delay jitter link traffic reliability controller load economic Expense	2018
	[17]	SSOA with chaotic maps	simulation: MATLAB. Hardware: Intel core i5 processor, Topology: Internet topology zoo	Time utilization cost utility	2019
	[18]	PSO	simulation: MATLAB. Hardware: ISP network with system 351 from Rocketfuel	SC and CC delay controller load imbalance number of controllers switch assignment	2017
	[19]	FFA	simulation: MATLAB. Hardware: Intel Core i5 with 4-Core processors	SC and CC latency multi-path connectivity between the switch and controller	2018
	[20]	APFO based MOCP schema	simulation: MATLAB. Topology: Internet2 OSE3, Tinet, RF-2	SC and CC and propagation latency Reliability load balance among controllers	2018
	[21]	VBO	simulation: MATLAB, python programming Topology: Abilene, Savvis, Biznet, Internet2 OS3E	Latency load balance (capitated and incapacitated)	2020
Genetic Algorithm Based CPP	[22]	Genetic Algorithm Hybridized by AHP technique	simulation: MATLAB. Hardware: 4-GH Intel Corei7 machine Topology: Internet topology zoo	Propagation delay hop count link utilization	2019
	[23]	Kuhn-Munkres algorithm Genetic algorithm	Hardware: Intel Core i7 (2.19 GHz) Topology: TA2, GERMANY50 and INDIA35	Propagation delay controllers load and balancing	2018
	[24]	MHNSGA	Simulation: MATLAB. Hardware: 4-GH Intel Corei7 machine Topology: Internet topology zoo	SC and CC latency Load balancing	2018
	[25]	MHNSGA-2	Simulation: MATLAB. Hardware: 4-GH Intel Corei7, CPU 4GH Topology: Internet topology zoo	SC and CC latency Load balancing	2018
	[26]	Genetic Algorithm (BIP)	Simulation: MATLAB. Hardware:2 Intel Xeon, E5–2430 processors with 8 cores, equipped with 16 GB of RAM Topology: Abilene, Janos-us, Pioro, Zib	Energy consumption Control paths delay controller load	2017
	[27]	MOGA (MHNSGA-2, PSO)	Topology: Internet topology zoo	load balancing delay	2021

controllers, and the estimated assignment cost on failure with minimal possible cost. In their technical paper, the Robust Dynamic Multi-Controller Placement (RDMCP) framework was integrated with a metaheuristic Particle Swarm Optimization (PSO) algorithm to develop a hybrid approach that improves the optimization of the objective function in terms of cost and reliability. The efficiency of their hybrid RDMCP-PSO was then benchmarked through comprehensive experimental evaluations and compared with a variety of benchmark algorithms. Results indicate that the proposed hybrid method significantly enhances network performance with several controllers employed and balancing loads compared to the standalone heuristic CPP algorithm.

Wu et al. suggested a CPP model for software-defined satellite networking (SDSN) to enhance performance using dynamic CPP (DCPP) and static CPP (SCPP) techniques [16]. Both techniques utilize a multi-objective optimization model solved by the APSO heuristic algorithm, with improved performance metrics compared to traditional SDSN, SCPP having lower economic costs, and DCPP offering greater flexibility.

Ateya et al. introduced a latency- and cost-conscious CPP scheme for large-scale SDN networks based on a dynamic optimization algorithm for the efficient use of controllers and distributed switches [17]. Their SSOA, improved with chaotic maps, showed enhanced performance in minimization of latency and deployment cost in simulations over game theory and meta-heuristic methods, yielding spectacular reliability

enhancement and computation efficiency.

Liao and Leung suggested a multi-objective genetic algorithm (MOGA) with a PSO-based mutation operator, which was found to be of improved performance in accuracy and convergence rate for single and multiple objectives in large search spaces [18].

Sahoo et al. proposed a Controller Placement Problem (CPP) strategy to minimize single link failures and at the same time minimize communication delays to enhance the performance of SDN [19]. They considered CPP as a multi-objective combinatorial optimization problem and utilized population-based meta-heuristics such as Particle Swarm Optimization (PSO) and Firefly Algorithm (FFA), which yielded optimal controller placement settings and switch-controller mappings. Simulations discovered notable average delay reductions under situations of single link failures.

In a recent paper, Zhang et al. proposed a complete CPP framework designed to optimize controller placement, switch-controller assignment, and the management ability of each controller in large-scale SDNs [20]. They formulated a multi-objective optimization controller placement (MOCP) problem and solved it using a bio-inspired heuristic algorithm known as Adaptive Bacterial Foraging Optimization (ABFO) that employed modified computational rules and chemotaxis. Performance assessment in three network topologies revealed that ABFO greatly improved reliability, reduced latency, and enhanced load balancing in comparison to the Particle Swarm Algorithm (PSA), with significant measures indicating higher reliability, load distribution, and reduced latency.

In a simultaneous study, Singh et al. proposed a heuristic CPP method for WAN based on SDN with significantly low average latency between controllers and switches thus improving overall SDN performance [21]. They proposed a novel optimization algorithm, Varna-Based Optimization (VBO), which best characterizes switches and controllers as dynamic particles, and proved its superiority in simulations on Matlab over famous algorithms such as PSO and Jaya.

2.2. GENETIC algorithm based CPP

The genetic algorithm evolution-based CPP planning schedules are elaborated fully in the second portion of Table 1, and this is followed by an extensive explanation.

Jalili et al. also specified a Switch to Controller Assignment Process that will enhance Quality of Service (QoS) in Software Defined Networks (SDN) based on an Analytic Hierarchy Process (AHP) for multi-criteria decision-making and optimal controller assignment [22]. Besides that, they also suggested a Genetic Algorithm for Controller Placement (GA-CP) to solve the Controller Placement Problem (CPP) with parameters such as controller load, latency, hop count, and link utilization, with simulations being more effective in load balancing than latency-based assignments.

Yuan et al. introduced the Controller Placement Problem (CPP) model efficiently and effectively to solve placement and assignment issues of Wide Area Networks (WAN) [23]. The model employs Genetic and Kuhn-Munkres algorithms for optimal solutions with minimum average propagation delay between switches and controllers. Experiments on various network topologies indicated significant performance enhancement in terms of propagation delay minimization and better load balancing, subject to the condition that CPP solutions are contingent upon other fundamental problems of the network such as resource scheduling and security.

Ahmadi and Khorramizade presented a multi-objective controller placement strategy with a global strategy in an efficient evolutionary algorithm adaptation, the Multi-Start Hybrid Non-Dominated Sorting Genetic Algorithm (MHNSGA), and performance metrics such as SC-latency, CC-latency, load balance, and reliability for link failure as a part of the objective function [24].

In subsequent research, Jalili et al. extended the MHNSGA algorithm [24] with a heuristic approach referred to as Multi-Start Hybrid NSGA-II (MHNSGA-II) to optimize resource allocation for large networks [25]. The extension constitutes a multi-objective optimization model capable of supporting measures of latency as well as load balancing using operations such as greedy initialization and an efficient Pareto finder to establish Pareto optimal solutions. Hu et al. employ a binary integer programming model blended with the genetic controller placement algorithm (IGCPA) to reduce the energy consumption of network control traffic with solutions very similar to the BIP with little extra link utilization [26].

Huang et al. presented an extensive framework for tackling the controller placement and scheduling problem (CPSP), amalgamating both challenges into a constrained optimization model that improves the efficiency of the control plane while reducing network latency [27]. The proficient resolution of the controller scheduling problem (CPS) was accomplished via a gradient-descent-based scheduling algorithm, which optimized the distribution of requests among controllers, while the controller placement problem (CPP) was addressed using a Clustering-based Genetic Algorithm with Cooperative Clusters (CGA-CC) to strategically position controllers within non-overlapping sub-networks and minimize the search space. Simulations illustrated that the GD-based algorithm resulted in a 37.5 % decrease in response time, while CGA-CC outperformed existing algorithms in expansive environments concerning computational costs, response times, and control plane efficacy.

3. Proposed method

To solve the problem of controller placement in the SDN network, we used a hybrid method in which we first examined the network nodes for selection as controller locations and attempted to sort the nodes based on their performance in the role of the network controller. Considering that various factors contribute to performance calculation and the impact of each factor on the overall network performance differs, we employed Data Envelopment Analysis (DEA) for this purpose. With this method, it is easy to select any number of controllers with the highest efficiency and priority from among the network nodes. The next issue is how to allocate switches to the selected controllers; to solve this problem, we also utilized the Green Anaconda Optimization method for the best allocations (Fig. 3). We will discuss the details of each stage of the proposed method in detail below.

3.1. First stage of the proposed method (Determining controller locations)

In the first stage, we use the Data Envelopment Analysis (DEA) method to select controller locations from among all the nodes in the SDN network. The goal of using this method is to select high-performance and optimal nodes as controllers, as the DEA method, due to its nature and mechanism, can easily calculate the performance of each node as a controller based on the characteristics of the network and

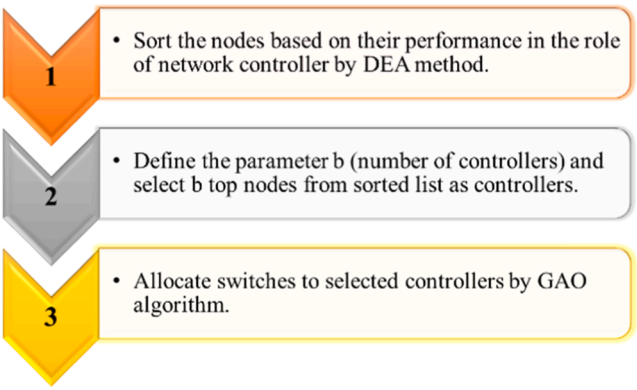


Fig. 3. High-level view of proposed DEA-GAO.

the parameters of each network node. Ultimately, we will have a sorted list of nodes based on their performance, allowing us to easily select the desired number of controllers for our network from the top of the list. We will first provide a brief explanation of the DEA method and then discuss how to use it in determining the optimal locations for the controllers.

3.1.1. Data envelopment analysis (DEA) and its application to optimization problems

Data Envelopment Analysis (DEA) constitutes a non-parametric technique employing mathematical programming to systematically assess the relative effectiveness of a collection of decision-making units that exhibit analogous inputs and outputs [10]. This article is a brief introduction to this technique developed first by Charnes, Cooper, and Rhodes all in 1978, now finding wide application with trends for using it growing among engineering, economics, management sciences as well as computer networks.

The DEA method shows the optimal Units of Efficiency Frontier, with all others below being individually inefficient relative points. Conversely, points further from this frontier are even less efficient in their own right and so on. The two main DEA models are [10]:

- CCR Modal

Assuming that the production function has constant returns to scale (CRS)–meaning changes in inputs will cause proportionate changes in outputs. The Charnes-Cooper-Rhodes model (CCR model for short) is structured based on fractional linear programming, attempting to maximize weighted inputs-to-output ratios as now determined for each decision-making unit [10].

- BCC Model

This model, developed by Banker, Charnes, and Cooper, considers variable returns to scale (VRS), allowing decision-making units to have increasing or decreasing returns under different conditions [10]. The BCC model models variable returns by adding a free variable to the CCR model. One of the advantages of the DEA method is that it makes the best decision for resource allocation and optimizing system performance solely based on its data, without the need for prior information about input-output functions. This characteristic has led us to leverage it to use it as a powerful tool in solving the problem of controller placement in SDN networks.

3.1.2. Determining the location of controllers using DEA method

The network performance in SDN networks is decided by optimal controller placement, and the Data Envelopment Analysis (DEA) method is a multi-criteria decision-making tool that enables us to compare the efficiency of various locations of controllers based on various criteria. Through this method, nodes that have the most suitable balance among various parameters can be selected as controllers. The biggest advantage of DEA compared to traditional methods is that it uses more than one effective criterion at a time instead of based on a single criterion and defines the relative efficiency of alternatives even without any pre-decided weights.

In this context, each node of a network is considered a DMU unit, and the BCC model of Data Envelopment Analysis (DEA) has been used to find the most effective controller placement in SDN networks. This model takes into account variable returns to scale, and since in an SDN network, a variation of one input parameter does not impact the output parameter proportionally, constant returns to scale hypothesis of the CCR model can lead to erroneous unit efficiency estimates. Here, BCC with greater flexibility and structural and parameter accounting scale differences is determined to be the suitable model for the node efficiency analysis and optimal controller placement. The mathematical form of this model is presented as follows.

$$\max_{\theta,\lambda} \theta \tag{1}$$

$$\sum_{j=1}^{n} x_j \lambda_{ij} \leq x_{i0}, \forall i$$

$$\sum_{j=1}^{n} y_j \lambda_{rj} \leq \theta y_{r0}, \forall r$$

$$\sum_{j=1}^{n} \lambda_j = 10 \lambda_j \geq, \forall j$$

Where:

- xij input value for node j for criteria that need to be minimized, such as the average distance from other nodes or network latency
- yrj output value for node j for criteria that need to be maximized, such as eigenvector centrality which indicates the influence of the node in the network
- λj weight of nodes in the convex combination to define node efficiency
- θ the efficiency level of the node under consideration
- The first constraints ensure that the inputs do not exceed the specified amount for the node in question.
- The second constraints ensure that the outputs are at least equal to the specified amount.
- The third constraint is specific to the BCC model and makes the return to scale variable (VRS).

Regarding the selection of parameters for the DEA method, we used graph structure centrality criteria [28] based on the SDN network structure. These criteria rank nodes based on their importance. Below, we briefly explain each of them:

i. Degree Centrality

Degree centrality constitutes one of the most fundamental and significant measures of centrality within the realm of network analysis. This metric quantifies the number of direct connections associated with a given node within the network and serves to illustrate the extent of connectivity of that node to other nodes within the network [28]. Degree centrality for a node i in an unweighted and undirected graph is formally delineated as follows:

$$C_D(i) = d(i) = |\{j \in (i,j) | V \in E\}| \tag{2}$$

where:

- $C_D(i)$ is the degree centrality value of node i.
- d(i) is the degree of node I (the number of edges connected to it).
- V is the set of nodes in the network.
- E is the set of edges in the network.

For weighted graphs, the degree of any node is usually calculated by the sum of the weights of all edges connected to that node.

$$C_D(i) = \sum_{j \epsilon N(i)} W_{ij} \tag{3}$$

It is where W_{ij} represents the weight of the edge connecting node i and node j.

Given the importance of selecting the best position for controllers in SDN networks, we use the degree of centrality criterion to place the controller because:

- Minimize communication delay: Typically, nodes with a higher degree of centrality are located in positions that afford them faster access to other nodes on the network.
- Increasing network reliability: Choosing high-degree nodes for controllers can help guard against network failure because those nodes are connected to many switches.
- Enhancing load system efficiency: Controllers placed on nodes with high-centrality degrees can bear a heavier workload this way and have more switches.

ii. Closeness Centrality

Closeness centrality is a measure of how close a node is to other nodes in the network [28]. Shorter paths mean faster access points among nodes, which can be calculated using this index.

The closeness centrality of a node i is defined by the formula (4):

$$C_c(i) = \frac{1}{\sum_j d(i,j)} \tag{4}$$

Where:

- CC(i) denotes the closeness centrality of node i.
- d(i,j) indicates the shortest distance between nodes i and j.

In a problem of controller placement for software-defined networks or a problem in a system with such requirements, for example, closeness centrality can serve as one of the criteria determining the optimal location for controllers. Controllers located near areas of high closeness are said to have quickly close communication with switches in the network and therefore low latency values.

iii. Betweenness Centrality:

Betweenness measures how important a node is to routing information through a network. This metric tells you how many times a node falls on the shortest path between pairs of other nodes [28]. The betweenness centrality of a node i is defined as follows:

$$C_B(i) = \sum_{s \neq i \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}} \tag{5}$$

Where:

- CB(i) is identified as the betweenness centrality attributed to node i.
- σst denotes the total count of shortest paths connecting nodes s and t.
- σst(i) represents the number of shortest paths between nodes s and t that traverse through node i.

Nodes with higher values of $C_B(i)$ are key points for network information transfer and they play a significant role in communication. Since a large amount of information flow passes through them, their positioning in a network can be critical for network efficiency. Nodes with high betweenness can form bottlenecks in a network, with serious consequences for efficiency.

In a problem of controller placement for software-defined networks or a problem in a system with such requirements, nodes with higher betweenness values are assumed to be the best places for locating new controllers. These nodes act like hubs of data flow within the network. Placing a controller in such nodes would, it is hoped, speed up communications and lead to improved network efficiency.

iv. Eigenvector Center Location

In the problem of controller placement in SDN networks, eigenvector centrality can be used as one of the criteria for determining the optimal location of controllers. It not only takes into account how many direct connections come out from this node (switch), but this standard also reflects the weight of connections that the connected vertex carries [28]. By this, if a switch is connected to some other more important and influential switches it will still have a sizeable degree of centrality. This means that if a switch is connected to more important and influential switches, it will sooner or later be more important itself. The eigenvector centrality of each switch i is defined as follows:

$$x_i = \frac{1}{\lambda} \sum_{j \in N(i)} x_i A_{ij} \tag{6}$$

where:

- xi is the eigenvector centrality value of switch i.
- Aij is the matrix entry at (i,j) in the adjacency matrix A, showing a connection or no connection between switches i and j.
- N(i) is the switch connected to i.
- λ is the largest eigenvalue of the matrix A.

Some switches have high eigenvector centrality values and are connected to influential nodes. Controllers located there naturally improve network performance. When you use this criterion in conjunction with other measures of centrality such as betweenness, closeness or node strength for finding the optimal location for controllers in a network, it is easiest and quickest because switches with high eigenvector centrality are already among the most important nodes in your graph.

v. Katz Centrality

Katz Centrality is one of the metrics for measuring the importance of nodes in a network that takes into account not only the number of direct connections (similar to degree centrality) but also indirect connections. This metric evaluates the significance of a node not only by the number of its immediate neighbors but also through its connections with other pivotal nodes in the network [28].

Katz Centrality for each node i is calculated as follows:

$$C_k(i) = \alpha \sum_j A_{ij} C_k(j) + \beta \tag{7}$$

where:

- $C_K(i)$ is the Katz centrality value for node i.
- A_{ij} is the element (i,j) in the adjacency matrix A (1 if there is an edge and 0 otherwise).
- α is a decay factor for the effect of longer paths (usually chosen to be less than the largest eigenvalue of matrix A to ensure convergence).
- β is a positive constant that ensures even nodes without direct connections have a positive centrality value.

This formula indicates that Katz's centrality considers not only the number of direct connections of a node but also accounts for indirect paths with a decay factor.

In the problem of controller placement in SDN, Katz centrality can help select nodes that not only have direct connections with other important nodes but also lie on key information paths. This feature allows controllers chosen based on this metric to have a high capability in managing network traffic and reducing latency.

vi. Information Centrality

Information Centrality is a measure of a node's importance in a network. It involves examining not just the length of this node and that path, but all ways that information can be transferred through paths connecting these two nodes as well. The assessment of the significance of a node within the network takes into account its role as a transfer point

for information between all other nodes in the net [28].

The information centrality of node i is calculated using the average transmission capability of information along paths between that node and other nodes. This metric is based on the inverse of the sum of path lengths between nodes, so nodes that have shorter paths to other nodes will be valued higher.

Information Centrality of node i in general can be calculated through the following formula:

$$C_I(i) = \sum_{i \neq j} d_{ij} \tag{8}$$

where:

- d_{ij} is the length of the path between nodes i and j.

You can see from this formula that how close node i is to other nodes in the net will greatly influence its information centrality.

In SDN networks, information centrality can be used to determine the best place for a controller. High-value nodes tend to be found in positions that allow for quick access to various parts of the net more efficiently spreading information around. This reduces latency that occurs when data is transferred between switches and controllers and therefore improves network performance generally.

vii. Influence Potential Centrality

Influence Potential Centrality is a network analysis metric that measures the ability of a node to influence other nodes. This metric takes into account the assumption that nodes located in strategic locations of the network would be capable of disseminating their information, ideas, or influence more quickly and widely in the network [28]. Influence Potential Centrality is typically derived from such measures as Eigenvector Centrality or Influence Spread Models. It is quantified based on the quantity and quality of the connections that a node has in the network.

One of the popular estimation approaches to Influence Potential Centrality applies the following function:

$$C_{IP}(i) = \sum_{j \epsilon N(i)} w_{ij} C_{IP}(j) \tag{9}$$

where:

- CIP(i) is the value of Influence Potential Centrality for node i.
- N(i) is the set of neighboring nodes of node i.
- wij is a weight that represents the degree of influence of node i on node j.

In SDN networks, the controller's optimal placement is very important as it has a direct impact on network performance. With the Influence Potential Centrality, the nodes that play the most significant role in information spreading in the network can be found.

This characteristic has the potential to minimize communication latency between switches and controllers and enhance network performance by choosing nodes that spread information extensively and rapidly, and also by optimizing control coverage and minimizing failure points in the network. Therefore, using Influence Potential Centrality in controller placement can lead to the design of a more efficient SDN network.

After sorting the network nodes based on their efficiency as controllers using the Data Envelopment Analysis method, it is time to allocate switches to the top k controllers on the sorted list, which we accomplished using the Green Anaconda optimization method, and we will explain this in detail in the next section.

3.2. Second stage of the proposed method (Allocating switches to selected controllers)

In the first stage of the proposed method, among all the network nodes, considering the graph centrality criteria, the efficiency values of the nodes were calculated as candidates for the controller locations using the DEA method, and they were sorted based on this. Now, by determining the number of controllers, we can easily select the nodes with high-efficiency values from the top of the sorted list for the final locations of the controllers. In the second stage of the proposed method, it is necessary to identify the switches that each controller will be responsible for managing. To this end, we use the Green Anaconda optimization method to allocate each switch from among the selected controllers to a controller in such a way that the overall productivity of the network reaches its highest level. In the following, while explaining the details of the GAO method, we will elaborate on the micro-operations of allocating switches to controllers.

In the year 2023, a novel metaheuristic algorithm known as Green Anaconda Optimization (GAO) was promulgated within the domain of optimization problem resolution, emulating the innate behavioral patterns of the green anaconda and has recently attracted considerable scholarly interest across a multitude of issues, particularly in the context of complex and nonlinear challenges [29]. This algorithm exhibits significant efficacy attributed to its innovative designs and distinctive characteristics in addressing optimization problems laden with numerous obstacles, particularly in intricate scenarios such as the allocation of controllers in Software-Defined Networking (SDN). The foundational inspiration for GAO is derived from the mechanisms employed by male green anacondas for locating material species during the reproductive season, in conjunction with the predation strategies exhibited by the species [29]. The mathematical formalization of GAO is predicated on the simulation of these dual strategies of the green anaconda, delineated into two phases: exploration and exploitation. The empirical results presented by the authors of the manuscript indicate that the GAO methodology possesses substantial capabilities in exploration, exploitation, and establishing a harmonious balance between optimization algorithms, outperforming its competing counterparts. In light of the aforementioned considerations, it was resolved that after the identification of controller locations via the Data Envelopment Analysis (DEA) method, the second phase would incorporate the green anaconda methodology, augmented by specific modifications for the allocation of switches to controllers. A comprehensive examination of the particulars associated with this phase will be conducted.

3.2.1. Initial setup of the algorithm

The Green Anaconda algorithm constitutes a population-based metaheuristic approach wherein the green anacondas function as the constituents of the population. From a mathematical perspective, each green anaconda is conceptualized as a candidate solution to the optimization problem, with its location within the search space representing the values of the decision variables [29]. Consequently, each green anaconda can be effectively represented through an array (Fig. 4). Given that in the initial stage of the proposed methodology, employing the DEA algorithm, out of n nodes within the network, b nodes have been designated as controller locations, it follows that there are n-$b = a$ nodes necessitating assignment to a controller. Hence, the dimensionality of the array symbolizing the anaconda must correspond to the total number of network switches, denoted as a, where each element of the array signifies a switch within the network, with its content encapsulating the index of the controller allocated to the respective switch (Fig. 4). Furthermore, the collective population of green anacondas, composed of

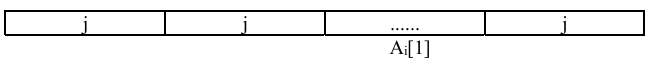


Fig. 4. Each population member (an anaconda).

these arrays, can be systematically represented through a matrix by as follows.

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_i \\ \vdots \\ A_N \end{bmatrix} N \times 1 = \begin{bmatrix} A_1[1] & \dots & A_1[t] & \dots & A_1[a] \\ \vdots & \ddots & & \iddots & \vdots \\ A_i[1] & \cdots & A_i[t] & \dots & A_i[a] \\ \vdots & \iddots & & \ddots & \vdots \\ A_N[1] & \dots & A_N[t] & \dots & A_N[a] \end{bmatrix} N \times a \tag{10}$$

Where:

- S: Set of switches S={S1,S2,…,Sa}
- C: Set of controllers C={C1,C2,…,Cb}
- a: Number of switches
- b: Number of controllers
- I: a member of the set {1,2,…,b}
- N: Number of anacondas
- Ai: An anaconda (a member of the population)
- A: Anacondas (population members) A={A1,A2,…,AN}

At the initiation of the algorithm's execution, it is imperative to randomly ascertain the initial position of each green anaconda within the search space. Since the content of each anaconda's variables determines its position in the search space, we must randomly assign each switch to a controller, and we do this using the formula (11).

$$Ai[t] = 1 + ri[t] * (b - 1) \tag{11}$$

where:

- i = 1,2,…,N
- t = 1,2,…,a

Based on the proposed values associated with each green anaconda for the decision variables, the objective function of the problem becomes evaluable. We defined the objective function as a combination of three parameters: the delay between the controller and the switch, the bandwidth, and the reliability of the path, with coefficients α, β, and γ.

$$Fitness(A_i) = Average\ [\alpha * Load\ Balance + \beta * Delay + \gamma * Reliability] \tag{12}$$

- Load Balancing

The load balancing parameter pertains to the equilibrium of loads across controllers as well as the load distribution among network links. As you know, controllers in the SDN network are responsible for managing and processing requests from switches. To examine the load balancing of controllers, we first need to calculate the load on each controller, namely the number of requests or the volume of traffic processed by the controller, and then obtain the load balancing of the controllers using the formula (13). It is evident that in this formula, the value of LBc close to zero indicates a uniform distribution of the load among the controllers.

$$LB_c = \sqrt[2]{\sum_{c=1}^{b} \left(\frac{1}{b} * (Load_c - \overline{Load})^2 \right)} \tag{13}$$

where:

- LBc: Load balancing of controllers in the network
- Loadc: A load of each controller (number of requests or volume of traffic processed).
- $\overline{load}$: Average load of Controllers
- b: Number of controllers

Now, if we want to examine the load balancing of links in the network, we first determine the load parameter of each link by calculating the number of packets that pass through the link per unit of time, and with the help of the formula (14), the load balancing of the links is obtained.

$$LB_L = \sqrt[2]{\sum_{l=1}^{z} \left(\frac{1}{z} * (Load_l - \overline{Load})^2 \right)} \tag{14}$$

where:

- LBL: Link Load Balancing in the Network
- LoadL: A load of each link (the number of packets transmitted over the link per unit of time).
- $\overline{load}$: Average load of Controllers
- z: Number of links

Finally, the load balancing parameter is obtained by averaging the link load balancing and the controller load balancing according to formula (15). In this paper, the impact of two factors, link load balancing and controller load balancing, on network efficiency is considered equal. If one of these two factors takes precedence over the other, their impact on the final load balancing calculation can easily be varied by applying coefficients.

$$Load\ Balance = (LB_L + LB_c)/2 \tag{15}$$

- Delay

To calculate the delay between the controller and the switch, we employ Dijkstra's algorithm to determine the shortest path connecting them, and then we use the delay of this path as the minimum delay between the controller and the switch in the objective function.

- Reliability

To calculate the reliability parameter, we also utilize formula (16), such that by multiplying the reliability of each link in the path between the controller and the switch and then averaging the reliability of all paths, the value of this parameter is obtained.

$$\text{Reliability} = \frac{\sum_{p=1}^{P} \left(R_path_p IR_path_p = \Pi_{l=1}^{L} R_lonk \right)}{P} \tag{16}$$

where:

- P: Number of paths between the controller and the switch
- L: Number of links in the path
- R_path: Path reliability
- R_link: Link reliability

Min-max normalization is used in this paper to scale the values of input variables with different ranges, i.e., delay, load balancing, and reliability, to a uniform range [0,1]. This is because scaling these values to a common range allows each variable an equal contribution to the overall fitness function. The general formula for Min-Max normalization is given as follows [30]:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{17}$$

Where:

- x represents the original value of the variable
- xmin and xmax denote the minimum and maximum values of the variable
- xnorm signifies the normalized value within the interval [0,1]

For variables where lower values signify superior performance (such as delay), an inverse normalization approach is implemented [30]:

$$x_{norm} = \frac{x_{max} - x}{x_{max} - x_{min}} \tag{18}$$

This guarantees that elevated normalized values consistently reflect enhanced fitness contributions across all evaluated criteria.

Now, after fully explaining how to calculate the values of the objective function, this set of values can be represented mathematically using a matrix according to formula (19).

$$F\begin{bmatrix} Fitness_1 \\ \vdots \\ Fitness_2 \\ \vdots \\ Fitness_n \end{bmatrix} N \times 1 = \begin{bmatrix} Fitness(A_1) \\ \vdots \\ Fitness(A_2) \\ \vdots \\ Fitness(A_N) \end{bmatrix} N \times 1 \tag{19}$$

where:

- F: Calculated objective function vector
- Fitness (Ai): The calculated objective function for the ith green anaconda
- Selection of coefficients for the objective function parameters

We used sensitivity analysis to evaluate the impact of the coefficients on the parameters of the objective function on the final results of the model. This method helps us find the optimal values for these coefficients and understand which parameter (delay, load balancing, and reliability) has the most significant impact on the output, and assigning a higher coefficient to that parameter leads to a more desirable result in the model output. For this purpose, in the simulation phase, we first selected initial values for the coefficients. In each execution cycle, we kept one coefficient constant and ran our algorithm multiple times for different values of the other two coefficients, recording the output results obtained each time, ultimately, by comparing the results, we stabilized the best values as the final values of the coefficients.

To ascertain the optimal placement of controllers within the Software-Defined Networking (SDN) framework, the individual corresponding to the highest objective value is designated as the optimal member (ideal candidate solution) through a comparative analysis of the computed values for the objective function. Given that in each iteration of the Green Anaconda Optimization (GAO) algorithm, the positions of the green anacondas, along with the corresponding objective function values, are recalibrated, the optimal member must be also revised accordingly.

3.2.2. Algorithmic mathematical model

In the context of the GAO design, the positioning of the green anacondas within the search domain is modified through the simulation of their behavior in two distinct phases aimed at both exploration and exploitation during the search procedure [29]. We will also try to develop a simplifying mathematical model here that will facilitate the assignment of switches to controllers in both steps of the GAO algorithm.

3.2.3. Exploration phase

The process of discovering the Green Anaconda Optimization method is based on the behavior of female green anacondas during the reproductive process in nature [29]. In this aspect, in the same way as the natural behavior of the female green anacondas, e.g., pheromonal secretion to attract male counterparts, the processes of this phase are performed [29]. At the initial phase of the GAO algorithm, which is the exploration stage, the position coordinates of green anacondas are modified through strategies used by male animals in the search for female mates and approaching them at the time of mating. This strategic approach engenders substantial alterations in the positional coordinates of green anacondas within the search space, thus underscoring the GAO's capacity for global exploration and precise examination of the problem-solving domain to circumvent entrapment within local optima [29]. In the context of the controller placement problem in Software-Defined Networking (SDN) architectures, each green anaconda delineates the status of switch allocation to controllers, indicating that the implementation of each designated anaconda as an array results in each cell corresponding to a network switch housing the identifier of the controller to which the switch is assigned. In essence, each anaconda symbolizes an articulated scenario for the configuration of the SDN network concerning the clustering of network switches and their allocation to controllers. Conversely, the value of the objective function associated with each anaconda signifies the efficacy of the scenario it proposes. Within this phase, for each anaconda, alternative anacondas that exhibit superior objective function values—thereby presenting more favorable network scenarios—are selected as female candidates. In these instances, the allocation of switches to controllers is orchestrated in such a manner that the overall proposed scenario yields a more advantageous outcome, thereby aiming to emulate enhanced scenarios to realize optimal switch-to-controller assignments at the network level. After the identification of the female candidate anacondas, one candidate is selected at random to facilitate the movement of the anaconda toward it. The movement of the anaconda towards the identified female anaconda in our framework signifies the emulation of the assignments proposed by the female anaconda, contingent upon the enhancement of the current state of the existing anaconda. The specifics of this procedural methodology will be expounded upon subsequently.

To mathematically represent this process, the design of the GAO assumes that for each green anaconda, the members of the GAO population that possess superior objective function values are designated as the female counterparts of the green anaconda. In other words, among all conceivable scenarios for the SDN network within the population space, the scenarios wherein the set of female candidate anacondas for each green anaconda is delineated are determined according to the formula (20).

$$CA(A_z) = \{A_y : Fitness_y > Fitness_z \text{ and } y \neq z\} \tag{20}$$

Where:

- y, z ∈ {1,2,…,N}
- CA(Az): the collection of female anacondas that qualify as candidates for the z-th anaconda.
- y: the index of the green anaconda within the GAO population matrix and the corresponding element's index in the objective function vector, which exhibits a superior objective function value relative to the z-th green anaconda.

In the natural environment, the secretion of pheromones by female anacondas plays a crucial role in attracting male counterparts, thereby influencing their movement towards the source [29]. To replicate the pheromone concentration, the objective function values, specifically the fitness values associated with each anaconda's proposed solution for the problem of switch allocation to controllers, are utilized. In essence, the more efficient the process of switch assignment to controllers is perceived to be for an anaconda, from the perspective of selective criteria, the greater the likelihood of that anaconda being recognized as a dominant female. Thus, within the operational framework of the method, a member's superior objective function value correlates positively with its likelihood of being selected by the green anaconda. The pheromone concentration probability function for the female species associated with each member of GAO is determined via formula (21).

$$Probability_{p.c.}(A_{u,v}) = \frac{Fitness(A_{u,v}) - Fitness(A_{u,max})}{\sum_{k=1}^{n_u}\{Fitness(A_{u,k}) - Fitness(A_{u,max})\}} \tag{21}$$

Where:

- u ∈ {1,2,…,N}

- v ∈ {1,2,…,nv}
- nu: the total count of female anaconda candidates for the u-th anaconda.
- Probabilityp.c. (Au,v): the likelihood of pheromone concentration of the v-th female anaconda relative to the u-th anaconda.
- Fitness(Au): an array encompassing the objective function values of the female anaconda candidates pertinent to the u-th anaconda.
- Fitness(Au,v): the objective function value corresponding to the y-th female anaconda candidate for the u-th anaconda.
- Fitness(Au,max): the maximum objective function value among the female anaconda candidates pertinent to the u-th anaconda.

In the GAO framework, it is posited that the green anaconda engages in a random selection of one candidate female anaconda and proceeds towards it. To emulate this selection mechanism, the cumulative probability function of the candidate female anacondas is initially computed through formula (22).

$$Cumulative_Probability(A_{u,v}) = Probability_{p.c.}(A_{u,v}) + Cumulative_Probability(A_{u,v-1}) \quad (22)$$

$$Cumulative_Probability(A_{u,1}) = 0$$

Where:

- u ϵ {1,2,…,N}
- v ϵ {2,…,nv}
- nv: the quantity of candidate female anacondas corresponding to the u-th anaconda.

Subsequently, by comparing the cumulative probability function values against a random variable (r) distributed normally within the interval [0,n], where n denotes the highest cumulative probability value of the female anacondas, the chosen female species for the green anaconda is ascertained following formula (23).

$$SA(u) = CA(u,v) : Cumulative_Probability(A_{u,v-1}) < r < Cumulative_Probability(A_{u,v}) \quad (23)$$

Where:

- SA(u): the identified female species for the u-th anaconda.

Upon the identification of the selected female anaconda and the simulation of the green anaconda's movement in its direction, a stochastic position within the search domain for the green anaconda is computed utilizing the formula (24). If the objective function value at this newly calculated position demonstrates an improvement, as indicated by formula (25), the position of the corresponding green anaconda is revised to this new location; otherwise, it retains its previous position.

$$A_{u-new}[t] = A_u[t] + r_u[t] * (SA_u[t] - I_u[t] * A_u[t]) \quad (24)$$

$$A_u = \begin{cases} A_{u-new} \text{ if } Fitness(A_{u-new}) > Fitness(A_u) \\ A_u \text{ , else} \end{cases} \quad (25)$$

Where:

- u ϵ {1,2,…,N}
- t ϵ {1,2,…,a}
- N: the total count of anacondas
- a: the total count of switches
- Au-new: the updated value about the u-th anaconda
- Au-new[t]: the corresponding controller associated with the t-th switch of the u-th anaconda.
- SAu[t]: the corresponding controller of the t-th switch within the selected female anaconda, relative to the u-th anaconda.
- Iu[t]: a stochastic value from the set {−1,+1}
- Fitness(Au-new): the value of the objective function for the newly assigned value of the u-th anaconda.

3.2.3.1. Exploitation phase. The subsequent phase of the GAO algorithm emulates the instinctual hunting behavior of the anaconda species. Consequently, the positions of the individuals within the population are recalibrated by the predatory strategies exhibited by the green anaconda [29]. This methodological approach results in marginal adjustments in the spatial positions of the green anacondas within the search domain, thereby demonstrating GAO's proficiency in capitalizing on localized searches for superior solutions in proximity to the identified solutions [29]. In the context of switch allocation to SDN network controllers, each anaconda epitomizes a specific networking scenario. During this phase, we implement a slight modification to the proposed scenario, which entails the reassignment of switches for each controller, with the stipulation of enhancing the overall network performance in terms of latency metrics, load distribution, and reliability.

To emulate the predatory strategy and the spatial realignment of the population members concerning the prey that has come within their vicinity, an initial random position is generated proximate to each green anaconda utilizing the formula (26). It is imperative to note that in this formula, our objective is to revise the value of each element within the anaconda array, which entails altering the controller designation such that the relevant switch in that specific element is allocated to the new controller. Subsequently, by formula (25), if the objective function value at this revised position exhibits an improvement, the modification of the green anaconda's position is regarded as valid.

$$A_{u-new}[t] = A_u[t] + (1 - 2 * r_u[t]) * \frac{b-1}{x} \quad (26)$$

Where:

- u ϵ {1,2,…,N}
- t ϵ {1,2,…,a}
- x: algorithm iteration counter and xϵ {1,2,…,T}
- N: total count of anacondas
- a: total count of switches
- b: total count of controllers
- T: maximum allowable iterations of the algorithm
- Au-new: updated value for the u-th anaconda
- Au-new[t]: newly assigned controller value corresponding to the t-th switch in the u-th anaconda.
- ru[t]: stochastic number within the range [0,1]

3.3. Overall algorithm framework

In the first stage of the algorithm, we receive information about an SDN network, including the number of nodes, the number of links, the delay of each link, the reliability of each link, the number of controllers, the capacity of the controllers, and the location of each controller, which is obtained in the first stage of the proposed method with the help of the DEA algorithm, as inputs to the problem. We then specify the population size and the number of iterations for the Green Anaconda optimization algorithm. To determine the number of iterations for the algorithm, various policies can be considered: reaching a specific value of the objective function, a fixed number of iterations, minor changes in the value of the objective function for several consecutive iterations, and execution time. After obtaining the initial information, we generate the population size, and an anaconda member with a random scenario for switch allocation to the controller (assigning each switch randomly to a controller).

Given that the number of switches under the control of each controller is limited and defined at the beginning of the algorithm, in every stage of the algorithm (initial population generation, exploration phase, or exploitation phase) where a change in the allocation of

switches to controllers occurs, the capacity of each controller must be checked before finalizing the new scenario. If the number of assigned switches exceeds the capacity threshold of the controller, we must apply a policy to adjust the scenario. In the proposed method, a function called "restoration function" is designed, which, upon receiving a network scenario, discovers overloaded controllers, meaning controllers with an overflow of switches, and adjusts them. The adjustment process involves randomly selecting switches from the set of allocated switches of that controller, equal to the number of excess switches, and assigning them to the nearest controller with available capacity.

In implementing this function, we utilized a delay matrix of dimensions b * b (where b is the number of controllers) that contains the delay between each pair of controllers. Another use of this matrix is during the determination of a backup controller. When the primary controller fails, all switches under its control are assigned to the nearest controller with available capacity using this delay matrix. After creating the initial population of the algorithm, at each iteration of the algorithm, first, in the exploration phase (simulating the mating of the green anaconda), each proposed scenario is modeled by one anaconda based on another anaconda's scenario with a better objective value, altering how it allocates its switches to the controller, and if necessary, the scenario is repaired and then updated. Then, in the exploitation phase (simulating prey hunting by the anaconda), a random location close to the member is created, and if the member's objective function improves, a transition to the new location occurs, meaning slight changes are made to the introduced scenario of each anaconda, provided that it improves its objective function, and the repair function is called again to ensure compliance with the controller thresholds. Finally, at the end of the iteration, we save the information of the best anaconda obtained by updating the best anaconda variable and proceed to another iteration of the algorithm. At the end of the last iteration of the algorithm, the best anaconda obtained is presented as the solution to the problem. Algorithm 1 represents the pseudocode of the problem, and Fig. 5 displays the overall flowchart of the proposed method.

Algorithm 1
Pseudocode switch assignment.

Start algorithm

1. Input network parameters: the number of nodes, the number of links, the delay and reliability of each link, the number of controllers, the capacity of controllers and the location of each controller.
2. Define the objective function.

$Fitness(A_i) = Average\ [\alpha * Load\ Balance + \beta * Delay + \gamma * Reliability]$

$Load\ Balance = (LB_L + LB_c)/2$

$$LB_c = \sqrt[2]{\sum_{c=1}^{b} \left(\frac{1}{b} * (Load_c - \overline{Load})^2\right)}$$

$$LB_L = \sqrt[2]{\sum_{l=1}^{z} \left(\frac{1}{z} * (Load_l - \overline{Load})^2\right)}$$

$$Reliability = \frac{\sum_{p=1}^{P} (R_path_p\ R_Ipath_p = \Pi_{l=1}^{L} R_link)}{P}$$

Delay: Calculated based on Dijkstra 's shortest path algorithm.

3. Set GAO population size (The number of different scenarios of each generation of the population) and iterations.
4. Generate the initial population (Basic random scenarios).

$A_i\ [t] = 1 + r_i[t] * (b - 1)$

5. For x = 1 to T
6. For i = 1 to N
7. Exploration phase:
8. Begin
9. Determine the candidate female anacondas (Scenarios with better objective function (.

$CA(A_z) = \{A_y : Fitness_y > Fitness_z\ and\ y \neq z\}$

10. Calculate the probability of pheromone concentration of candidate anacondas.

$$Probability_{p.c.}(A_{u,v}) = \frac{Fitness(A_{u,v}) - Fitness(A_{u,\max})}{\sum_{k=1}^{n_u} \{Fitness(A_{u,k}) - Fitness(A_{u,\max})\}}$$

11. Calculate cumulative probability of candidate anacondas.

$Cumulative_Probability(A_{u,v}) = Probability_{p.c.}(A_{u,v}) + Cumulative_Probability(A_{u,v-1})$

$Cumulative_Probability(A_{u,1}) = 0$

12. Get selected female anaconda randomly.

SA(u)=CA(u,v): Cumulative_Probability($A_{u,v-1}$) < r < Cumulative_Probability($A_{u,v}$)

13. Choose a new location to move to the selected Anaconda.

$A_{u-new}[t] = A_u[t] + r_u[t] * (SA_u[t] - I_u[t] * A_u[t])$

14. Calculate objective function of new location.
15. Update ith anaconda scenario.

$$A_u = \begin{cases} A_{u-new}\ if\ Fitness(A_{u-new}) > Fitness(A_u) \\ A_u\ ,\ else \end{cases}$$

16. Repair ith anaconda scenario.
17. End
18. Exploitation phase:
19. Begin
20. Calculate the new location for ith anaconda.

$$A_{u-new}[t] = A_u[t] + (1 - 2 * r_u[t]) * \frac{b-1}{x}$$

21. Update ith anaconda scenario.

$$A_u = \begin{cases} A_{u-new}\ if\ Fitness(A_{u-new}) > Fitness(A_u) \\ A_u\ ,\ else \end{cases}$$

22. Repair ith anaconda scenario.
23. End
17. End.
18. Save the best scenario
19. End.
20. Output the obtained best scenario.

End algorithm

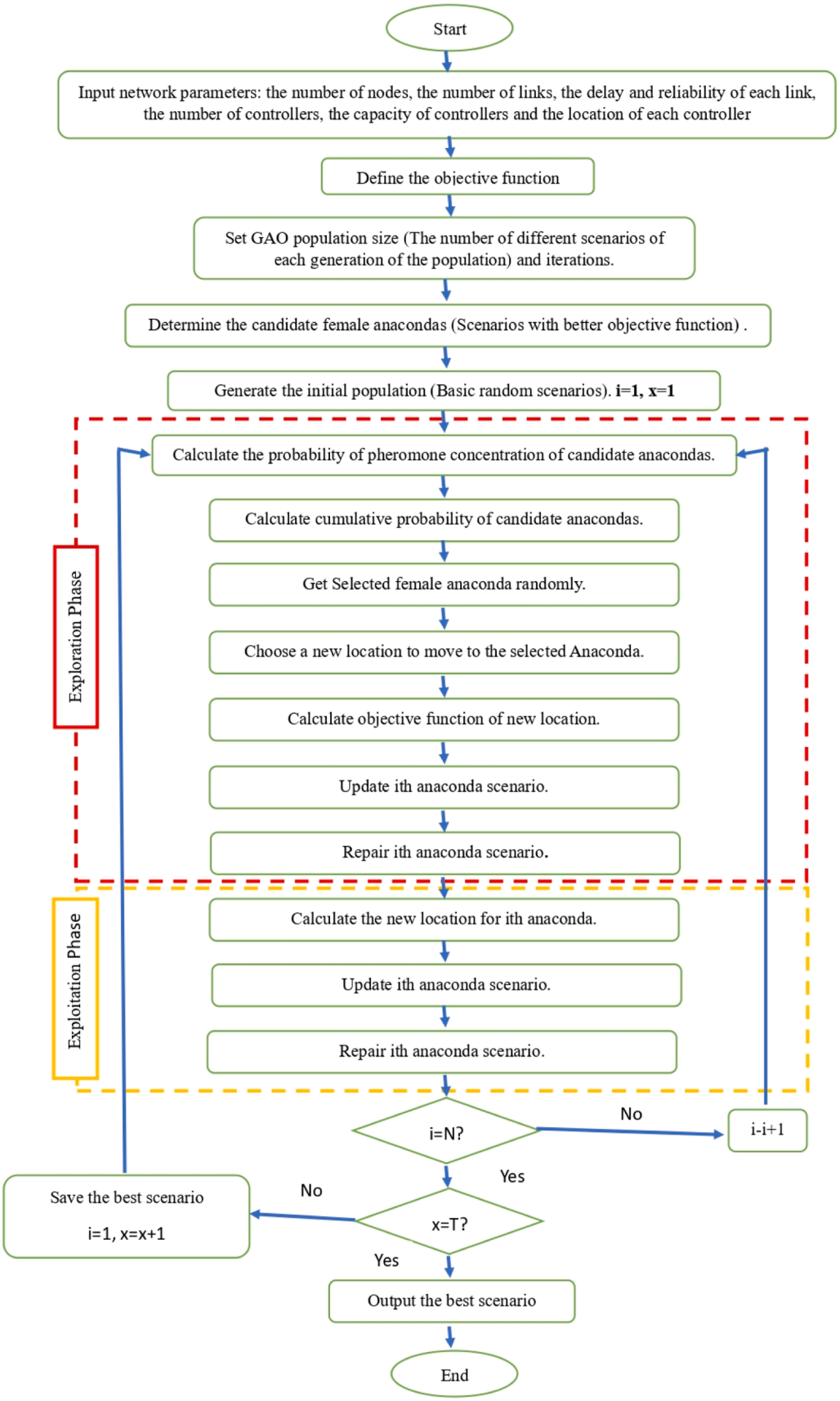


Fig. 5. The overall flowchart of the proposed method.

4. Performance evaluation and results

This section first describes the detailed simulation settings in Section 4.1 and then presents the simulation results of the proposed method in Section 4.2 and finally evaluates the performance of our proposal with the previous three related methods in several metrics in Section 4.3.

4.1. Simulation environment

Our simulation framework is established on a physical server (Intel (R) Xeon(R) Platinum 8180, CPU 2.50 GHz, Ubuntu 20.04, 64 GB RAM). In addition, the Python programming language is employed to execute the proposed algorithms. The development environment utilized is Python 3.13, with a primary reliance on the numpy and matplotlib libraries, alongside the ONOS controller throughout the simulation processes.

4.2. Topology information

The simulation and planning model are executed and assessed across six topologies selected from the Internet Topology Zoo [31]. These topologies include Aarnet, Agis, USnet, Chinanet, Bellcanada, and Iris_net (refer to Table 2). To affirm the efficacy and optimization of the proposed methodology across diverse scenarios, these topologies were selected based on their size, ranging from a limited number of nodes to those incorporating a substantial quantity of nodes.

4.3. Simulation parameters

The network parameter values are derived from [15], featuring a processing controller capacity of 7.8 k request packets/s. The demand requested spans from [100 to 400] k req/s, while the access bandwidth is set at 10 Gbps. The proposed algorithm is executed for 150 iterations, initiating with a population of 50 members, and the placement scheme yielding optimal performance is subsequently selected.

4.4. Simulation comparison

The proposed algorithm underwent comparison with three distinct methodologies: the original Particle Swarm Optimization (PSO) algorithm as referenced in [32], a hybrid RDMCP-PSO algorithm drawn from [15], and the heuristic CPP algorithm documented in the research presented in [33].

4.5. Simulation results

In the simulation, each network under study is defined as a graph, with the network nodes as graph nodes and the links between the network nodes as graph edges. Centrality measures are calculated for the entire graph, and based on these measures and with the help of the DEA method, the nodes are ranked as candidates for being controllers. For each execution of the second stage of the algorithm, which is the assignment of other nodes to controllers, the number of controllers is first determined, and the specified number of high-performance nodes obtained in the first stage are selected as controllers. Then, we utilized the GAO algorithm to allocate other nodes to the selected controllers. This algorithm has two phases: exploration and exploitation, and during each phase, the population members (anacondas), each representing a network scenario, attempt to improve their fitness by moving toward the optimal location in the solution space.

Table 2
Network data information.

Network name	nodes	edges
Aarnet	19	24
Agis	25	30
Usnet	139	195
Chinanet	42	66
Bellcanada	48	65
Iris_net	102	64

In the simulation, we monitored and tracked the number of movements of the anacondas, each of which represents an attempt to find a solution to the controller placement problem, and we obtained the results presented in Table 3. This table contains simulation information for different numbers of controllers and executions, as well as for various network topologies. In each execution, a fixed number of generations, equal to 150 generations, was considered, and for each network topology, the average number of anaconda movements was calculated and recorded. These movements occurred only if the new location of the anaconda improved compared to its current location, meaning that better performance was achieved in the resulting scenario. Some of the resulting scenarios were repetitive, meaning they were produced in the current generation or previous generations, while others were new solutions for the problem of allocating switches to controllers.

Fig. 6 displays the average number of new scenarios obtained during various executions in the studied networks.

As you know, the movements of the anacondas and consequently the generation of new network scenarios with a better objective function value ultimately led to the improvement of network parameters such as average delay, and this improvement trend is observable in Fig. 7.

4.6. Performance evaluation

The goal of our algorithm is to provide an optimal scenario for the SDN network with the best controller placement and the best assignment of switches to controllers to achieve the lowest latency, the highest load balance, and the highest reliability and the solution that has been devised to failures (links and nodes) in the network is to assign the node to the closest controller with capacity using the Dijkstra algorithm. In the simulation phase, the system components only need to be tested against failures during operation. That means the overload of frequent control failures relates proportionally to the overall estimation of reliability. A comparison is made between the four algorithms: PSO [32], RDMCP-PSO [15]f, heuristic CPP [33], and our proposed DEA-GAO, to show the average delay, controller number, and utilization of various networks.

The reliability results highlighted two instances, namely before and after the controller's failure for each algorithm (illustrated in Fig. 8). The average latencies exhibited a notable decrease as the number of controllers increased. Fig. 8-a illustrates the average latency preceding the failure of the reliable link across various networks. The DEA-GAO demonstrates the lowest average delay among these networks. It becomes clear that, for each network, the delay is contingent upon the number of nodes and their respective distances from the controllers. Consequently, the introduction of an additional controller may facilitate a reduction in delay, contingent upon the network's scale. In summary, among the six assessed topologies, the DEA-GAO methodology demonstrates a 28.8 % enhancement relative to the Heuristic CPP approach, a 22.2 % enhancement compared to the PSO methodology, and an 8.8 % enhancement about the RDMCP-PSO approach.

Conversely, Fig. 8-b depicts the average delay after failure for the same networks. Therefore, in terms of reliability, the overall delay escalates across all four methodologies due to diminished resource scheduling efficiency. Furthermore, the latency following failure is elevated as switches allocate time to reacquire access to the subsequent controller. Note that our algorithm performs better than the others in terms of latency and it has an improvement of 30.7 % over the Heuristic CPP method, 21.5 % over the PSO method, and 2.9 % over the RDMCP-PSO method.

The load balancing and utilization of SDN networks achieved by different algorithms are shown in Fig. 9. The utilization of the

Table 3
Average of anaconda moves during different phases of the GAO algorithm.

# iteration	Exploration Phase						Exploitation Phase					
	Network name						Network name					
	Aarnet	Agis	Bellcanada	Chinanet	Iris_net	Usnet	Aarnet	Agis	Bellcanada	Chinanet	Iris_net	Usnet
1–10	64	91	169	122	179	242	25	40	82	77	94	122
11–20	62	89	166	121	175	237	23	35	73	69	83	109
21–30	44	63	117	83	123	166	27	42	85	77	96	126
31–40	55	79	145	105	154	208	28	44	90	81	101	132
41–50	45	64	117	83	123	167	29	45	92	83	103	136
51–60	33	48	87	62	92	125	18	28	57	53	65	85
61–70	26	0	69	50	73	98	8	13	27	26	31	40
71–80	35	50	92	68	98	132	6	10	23	24	27	35
81–90	31	45	83	61	88	119	3	6	14	16	17	22
91–100	22	31	58	43	62	84	4	6	13	14	15	20
101–110	16	23	42	30	44	60	3	5	12	12	14	18
111–120	10	14	26	19	28	37	3	5	11	10	12	16
121–130	5	8	14	0	15	20	2	3	7	0	8	11
131–140	6	0	0	0	0	0	4	0	0	0	0	0
141–150	0	0	0	0	0	0	0	0	0	0	0	0

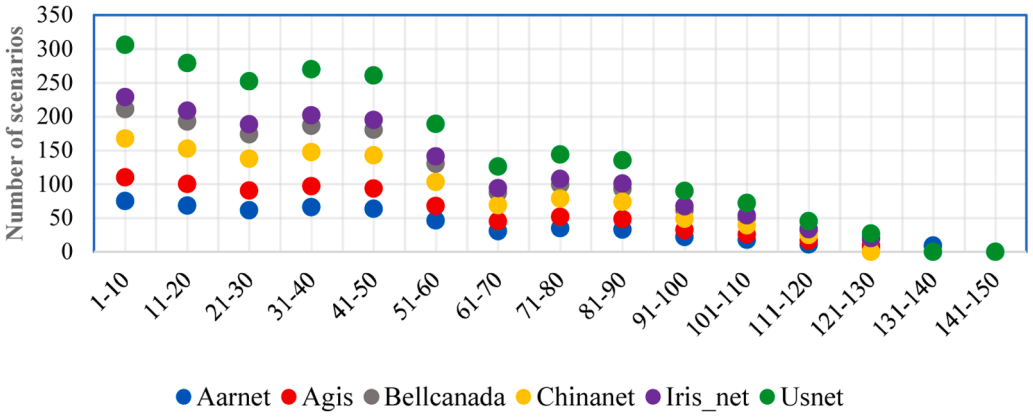


Fig. 6. Average of new scenarios created per population generation.

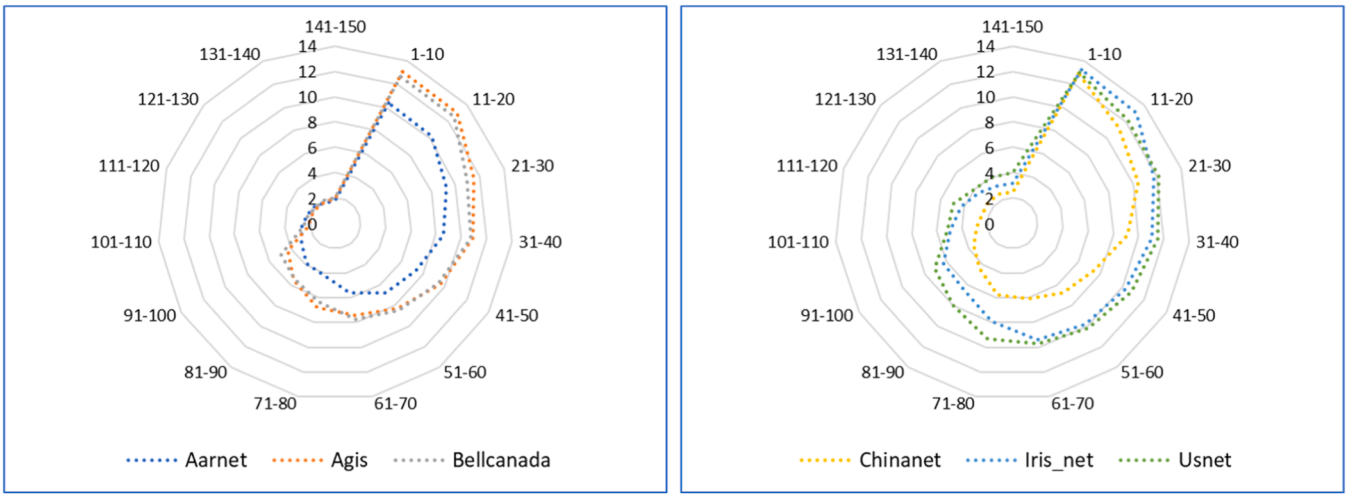
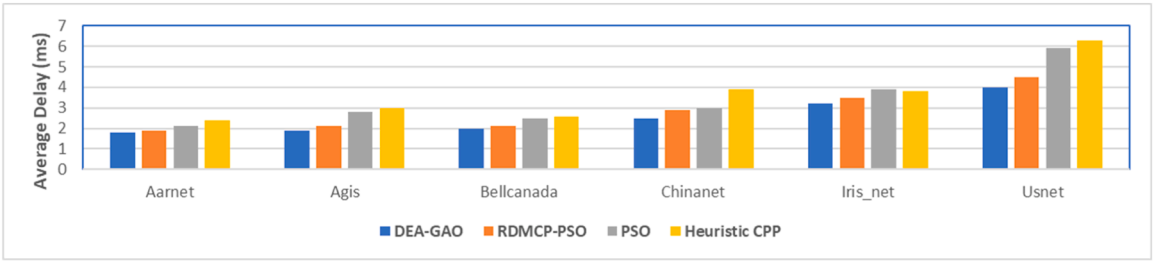


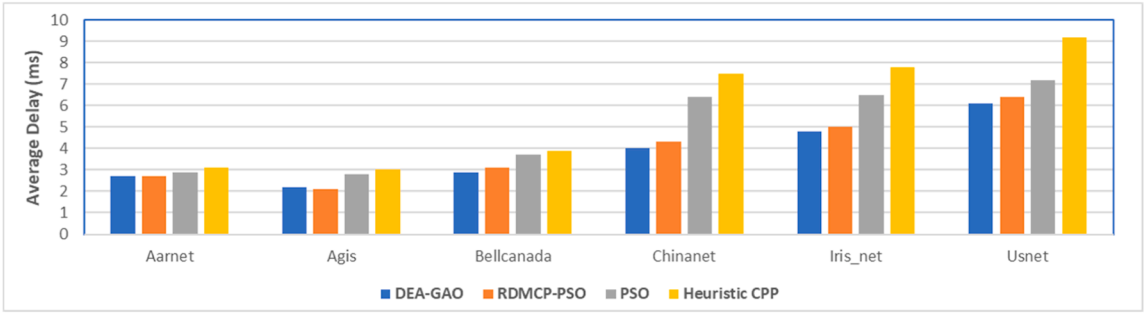
Fig. 7. The improvement trend of the average delay metric over different generations.

controller's index in normal operation reflects the network performance or efficiency of the resource scheduling for all controllers with the set of switches and it is obtained by dividing the total request rate of each controller's switches by the total processing capacity of the controllers. The findings indicate that the suggested approach demonstrates enhancements of 7.3 %, 32 %, and 1.5 % in comparison to the Heuristic CPP, PSO, and RDMCP-PSO methodologies, respectively, across a comprehensive assessment of 6 distinct topologies.

As a concluding evaluation, the optimal number of controllers across all four methodologies is compared and depicted in Fig. 10. Different parameters have been assigned to the controllers for these algorithms. It is observed that the selection of controllers varies according to the node



a) before failure



b) after failure

Fig. 8. The average delay over various topologies.

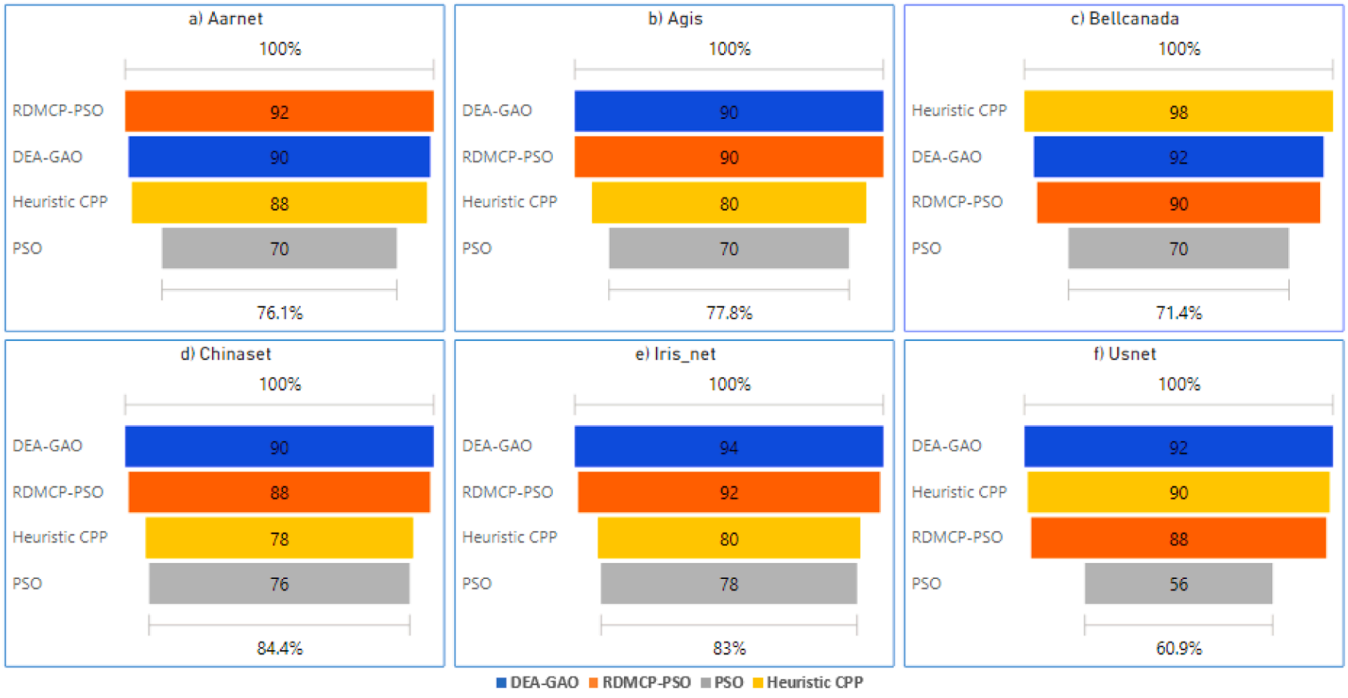


Fig. 9. The optimal controller utilization.

density within each network, which is influenced by the precision of the load-balancing mechanism. The DEA-GAO is capable of minimizing deployment costs between switches and controllers while incurring a low delay cost. It guarantees an equitable distribution of controllers throughout the network while simultaneously optimizing resource allocation to attain a satisfactory level of reliability.

5. Conclusion and future work

Determining the optimal quantity of controllers and the ideal locations for their installation represent significant challenges within the multi-controller model, collectively referred to as the Controller Placement Problem (CPP). The CPP constitutes an optimization dilemma aimed at minimizing latency while achieving a balanced network configuration. The significance of CPP is accentuated when the objectives include enhancing reliability and diminishing delay. In this study, a two-stage methodology for the strategic placement of controllers and the allocation of switches in Software-Defined Networking (SDN) environments was articulated. In the initial stage, the BCC model of Data Envelopment Analysis (DEA) was employed in conjunction with graph

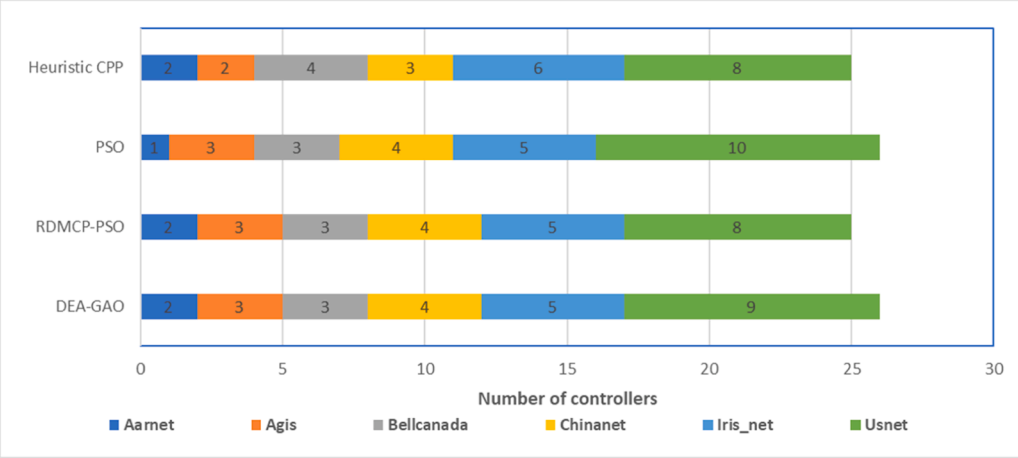


Fig. 10. The optimal number of controllers.

centrality metrics to ascertain nodes exhibiting the highest efficiency for controller deployment. Subsequently, in the second stage, the optimal allocation of switches to controllers was executed utilizing the Green Anaconda Optimization algorithm. The findings indicate that the proposed hybrid approach (DEA-GAO) markedly enhances network performance in terms of average latency and controller utilization compared to the three preceding methodologies (The average improvement in the delay parameter compared to previous methods is 20 % and the average improvement in the controller utilization parameter compared to previous methods is 13.6 %.).

In prospective research endeavors, the evolution of the proposed model will encompass dynamic network variables such as fluctuations in traffic, node failures, and load migration. Furthermore, the integration of the DEA approach with machine learning algorithms is anticipated to substantially enhance decision-making precision and adaptability about actual network conditions, which constitutes one of the authors' objectives in the continuation of their research.

CRediT authorship contribution statement

Fatemeh Dashti: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ali Ghaffari:** Writing – review & editing, Validation, Supervision, Conceptualization. **Nahideh Derakhshanfard:** Writing – review & editing, Writing – original draft, Supervision. **Shiva Taghipoureivazi:** Writing – review & editing, Writing – original draft, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] B. Sapkota, A. Ray, M.K. Yadav, B.R. Dawadi, S.R. Joshi, Machine learning-based attack detection and mitigation with multi-controller placement optimization over SDN environment, J. Cybersecur. Priv. 5 (1) (2025 Mar 19) 10.
[2] A.M. Abdulghani, A. Abdullah, A.R. Rahiman, N.A. Hamid, B.O. Akram, H. Raissouli, Navigating the Complexities of Controller Placement in SD-WANs: A Multi-Objective Perspective on Current Trends and Future Challenges, Computer Systems Science & Engineering, 2025 Jan 1, p. 49.
[3] M.A. Ouamri, T. Alharbi, D. Singh, Z. Sylia, A comprehensive survey on software-defined wide area network (SD-WAN): principles, opportunities and future challenges, J. Supercomput. 81 (1) (2025 Jan) 1–47.
[4] T. Darwish, T.A. Alhaj, F.A. Elhaj, Controller placement in software defined emerging networks: a review and future directions, Telecommun. Syst. 88 (1) (2025 Mar) 1–33.
[5] R. Chaudhary, G.S. Aujla, N. Kumar, P.K. Chouhan, A comprehensive survey on software-defined networking for smart communities, Int. J. Commun. Syst. 38 (1) (2025 Jan 10) e5296.
[6] M. Kolli, M. Karre, A comprehensive study on optimization algorithms for controller placement in software-defined networking, in: 2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), IEEE, 2025 Feb 5, pp. 596–602.
[7] Q. Wu, J. Fang, J. Zeng, J. Wen, F. Luo, Monte Carlo simulation-based robust workflow scheduling for spot instances in cloud environments, Tsinghua Sci. Technol. 29 (1) (2023 Aug 21) 112–126.
[8] Y. Shi, X. Hou, Z. Na, J. Zhou, N. Yu, S. Liu, L. Xin, G. Gao, Y. Liu, Bio-inspired attachment mechanism of dynastes hercules: vertical climbing for on-orbit assembly legged robots, J. Bionic. Eng. 21 (1) (2024 Jan) 137–148.
[9] K. Li, L. Ji, S. Yang, H. Li, X. Liao, Couple-group consensus of cooperative–competitive heterogeneous multiagent systems: a fully distributed event-triggered and pinning control method, IEEe Trans. Cybern. 52 (6) (2020 Oct 15) 4907–4915.
[10] A. Emrouznejad, E. Cabanda, Introduction to Data Envelopment Analysis and Its applications. InHandbook of Research On Strategic Performance Management and Measurement Using Data Envelopment Analysis, IGI Global Scientific Publishing, 2014, pp. 235–255.
[11] V.S. Thalapala, K.WOCP Guravaiah, Controller placement using whale optimization in SDN-WAN, Procedia Comput. Sci. 252 (2025 Jan 1) 883–892.
[12] G.D. Singh, V. Tripathi, A. Dumka, R.S. Rathore, M. Bajaj, J. Escorcia-Gutierrez, N. O. Aljehane, V. Blazek, L. Prokop, A novel framework for capacitated SDN controller placement: balancing latency and reliability with PSO algorithm, Alex. Eng. J. 87 (2024 Jan 1) 77–92.
[13] C. Li, K. Jiang, Y. Luo, Dynamic placement of multiple controllers based on SDN and allocation of computational resources based on heuristic ant colony algorithm, Knowl.-Based Syst. 241 (2022 Apr 6) 108330.
[14] S. Tahmasebi, N. Rasouli, A.H. Kashefi, E. Rezabeyk, H.R. Faragardi, SYNCOP: an evolutionary multi-objective placement of SDN controllers for optimizing cost and network performance in WSNs, Comput. Netw. 185 (2021 Feb 11) 107727.
[15] A.A. Ibrahim, F. Hashim, A. Sali, N.K. Noordin, K. Navaie, S.M. Fadul, Reliability-aware swarm based multi-objective optimization for controller placement in distributed SDN architecture, Digit. Commun. Netw. 10 (5) (2024 Oct 1) 1245–1257.
[16] S. Wu, X. Chen, L. Yang, C. Fan, Y. Zhao, Dynamic and static controller placement in software-defined satellite networking, Acta Astronaut. 152 (2018 Nov 1) 49–58.
[17] A.A. Ateya, A. Muthanna, A. Vybornova, A.D. Algarni, A. Abuarqoub, Y. Koucheryavy, A. Koucheryavy, Chaotic salp swarm algorithm for SDN multi-controller networks, Eng. Sci. Technol. Int. J. 22 (4) (2019 Aug 1) 1001–1012.
[18] L. Liao, V.C. Leung, Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in SDNs, in: 2017 IEEE conference on network function virtualization and software defined networks (NFV-SDN), IEEE, 2017 Nov 6, pp. 1–6.
[19] K.S. Sahoo, D. Puthal, M.S. Obaidat, A. Sarkar, S.K. Mishra, B. Sahoo, On the placement of controllers in software-defined-WAN using meta-heuristic approach, J. Syst. Softw. 145 (2018 Nov 1) 180–194.
[20] B. Zhang, X. Wang, M. Huang, Multi-objective optimization controller placement problem in internet-oriented software defined network, Comput. Commun. 123 (2018 Jun 1) 24–35.

[21] A.K. Singh, S. Maurya, S. Srivastava, Varna-based optimization: a novel method for capacitated controller placement problem in SDN, Front. Comput. Sci. 14 (2020 Jun) 1–26.
[22] A. Jalili, M. Keshtgari, R. Akbari, R. Javidan, Multi criteria analysis of controller placement problem in software defined networks, Comput. Commun. 133 (2019 Jan 1) 115–128.
[23] T. Yuan, X. Huang, M. Ma, J. Yuan, Balance-based SDN controller placement and assignment with minimum weight matching, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018 May 20, pp. 1–6.
[24] V. Ahmadi, M. Khorramizadeh, An adaptive heuristic for multi-objective controller placement in software-defined networks, Comput. Electr. Eng. 66 (2018 Feb 1) 204–228.
[25] A. Jalili, M. Keshtgari, R. Akbari, Optimal controller placement in large scale software defined networks based on modified NSGA-II, Appl. Intell. 48 (2018 Sep) 2809–2823.
[26] Y. Hu, T. Luo, N.C. Beaulieu, C. Deng, The energy-aware controller placement problem in software defined networks, IEEe Commun. Lett. 21 (4) (2016 Dec 28) 741–744.
[27] V. Huang, G. Chen, P. Zhang, H. Li, C. Hu, T. Pan, Q. Fu, A scalable approach to SDN control plane management: high utilization comes with low latency, IEEE Trans. Netw. Serv. Manag. 17 (2) (2020 Feb 11) 682–695.
[28] N. Meshcheryakova, S. Shvydun, A comparative analysis of centrality measures in complex networks, Autom. Remote Control 85 (8) (2024 Aug) 685–695.
[29] M. Dehghani, P. Trojovský, O.P. Malik, Green anaconda optimization: a new bio-inspired metaheuristic algorithm for solving optimization problems, Biomimetics. 8 (1) (2023 Mar 14) 121.
[30] Han Y., Jia Z., He S., Zhang Y., Wu Q. CNN+ Transformer based anomaly traffic detection in uav networks for emergency rescue. arXiv prepr. arXiv:2503.20355. 2025 Mar 26.
[31] Internet topology zoo, http://www.topology-zoo.org/.
[32] L. Liao, V.C. Leung, Z. Li, H.C. Chao, Genetic algorithms with variant particle swarm optimization based mutation for generic controller placement in software-defined networks, Symmetry 13 (7) (2021 Jun 24) 1133.
[33] A.A. Ibrahim, F. Hashim, N.K. Noordin, A. Sali, K. Navaie, S.M. Fadul, Heuristic resource allocation algorithm for controller placement in multi-control 5G based on SDN/NFV architecture, IEEe Access. 9 (2020 Dec 24) 2602–2617.