Journal of Network and Computer Applications 225 (2024) 103869

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Research paper

A Stochastic Computational Graph with Ensemble Learning Model for solving Controller Placement Problem in Software-Defined Wide Area Networks

Oladipupo Adekoya *, Adel Aneiba

Birmingham City University, Birmingham, United Kingdom

ARTICLE INFO

Keywords:
Computational graph
Controller placement
SDWAN
XGBoost

ABSTRACT

The Preponderance of literature has established that most of the metaheuristic algorithms were associated with identified challenges in solving the Controller Placement Problem in SD-WAN. This study proposed a Stochastic Computational Graph Model with an Ensemble Learning (SCGMEL) approach to address the scalability, intelligence, and high computational complexity challenges experienced by the existing metaheuristic algorithms. The proposed SCGMEL used stochastic gradient descent with momentum and learning rate decay, a computational graph model, and the eXtreme Gradient Boosted Trees (XGBoost) algorithm as the optimization and machine learning approaches. The proposed solution was tested using datasets from Internet Zoo topology with six objective functions: load balancing, maximum controller failure, average controller-to-controller latency, average switch-to-controller latency, and maximum controller-to-controller latency. The XGBoost outperformed other regression models, in predicting the number of controllers, with mean absolute error of **1.855751** versus **1.883536**, **3.729863**, and **3.829268** for the random forest, logistic regression, and K-nearest neighbor, respectively. Furthermore, the execution time, average and total CPU usages of the algorithms demonstrated the computational efficiency of the proposed SCGMEL over ANSGA-III, NSGA-II, and MOPSO with percentage decreases of **99.983%**, **99.985%**, and **99.446%**, respectively. Consequently, the proposed SCGMEL was recommended for controller placement in SD-WAN, subject to the usage conditions.

1. Introduction

A fundamental change in the way that communication networks operate has been catalyzed by the dissemination of Software-Defined Networking, also known as SDN (Wang et al., 2017). This shift is directed towards a logically centralized architecture where the control plane and data plane are segregated. This design facilitates the relocation of control operations from network devices to specialized software-based controller instances situated centrally (Liu et al., 2015). The centralized controller assumes responsibility for managing network switches and establishing forwarding rules that dictate packet behavior (Tootoonchian and Ganjali, 2010). While the controller's logical presence is centralized, its spatial distribution is essential to address factors like fault tolerance, scalability, resilience, and performance (Ros and Ruiz, 2016).

In the realm of Wide Area Networks (WAN), the conventional centralized approach faces limitations, particularly in terms of performance and scalability. Addressing these challenges entails the deployment of multiple controllers to collaboratively oversee the network, a task known as controller placement (CP) (Heller et al., 2012). The strategic identification of a limited number of controllers meeting specific conditions becomes pivotal in Software-Defined Wide Area Networks (SD-WAN) (Hock et al., 2013). Randomly determining the number and location of controllers does not guarantee satisfactory performance. Therefore, to achieve an effective and dependable SD-WAN, questions surrounding the optimal number and location of controllers must be resolved (Heller et al., 2012).

Traditional approaches, such as exhaustive methods or metaheuristic techniques, grapple with computational complexity, lack of predictive intelligence, and scalability issues when addressing the controller placement problem (Adekoya and Aneiba, 2022). While mathematical optimization methods like CPLEX optimizer and integer linear programming offer optimality, they struggle with computational costs and local optima entrapment (Giroire et al., 2014). Metaheuristics like Multi-Objective Particle Swarm Optimization (MOPSO), Nondominated Sorting Genetic Algorithm II (NSGA-II), and Adapted Nondominated Sorting Genetic Algorithm III (ANSGA-III) are utilized as computational tools or methodologies to guide the search for optimal or near-optimal solutions for controller placement within the SD-WAN network (Gao et al., 2015; Ahmadi et al., 2015; Adekoya and Aneiba, 2022). However, it is important to note that, apart from ANSGA-III, these approaches grapple

* Corresponding author.
E-mail address: oladipupo.adekoya@mail.bcu.ac.uk (O. Adekoya).

https://doi.org/10.1016/j.jnca.2024.103869
Received 5 March 2023; Received in revised form 29 January 2024; Accepted 12 March 2024
Available online 20 March 2024

with scalability challenges. Moreover, they fall short in predicting the required number of controllers and tend to be computationally expensive when compared to the innovative approach proposed in this study.

This study introduces a groundbreaking solution, the Stochastic Computational Graph Model with Ensemble Learning (SCGMEL), with the primary aim of overcoming the substantial challenges that existing metaheuristic algorithms encounter when addressing the Controller Placement Problem within SD-WAN. Departing from previous methodologies, the SCGMEL method unveils an exceptional innovation by seamlessly integrating mini-batch stochastic gradient descent with momentum and learning rate decay, a computational graph model, and the powerful eXtreme Gradient Boosted Trees (XGBoost) algorithm.

Meticulously crafted, the SCGMEL approach is designed to directly confront three pivotal challenges: scalability, intelligence, and the intricate computational complexity that have hitherto presented significant obstacles for existing metaheuristic algorithms seeking to solve the Controller Placement Problem within SD-WAN. Through the synergistic incorporation of mini-batch stochastic gradient descent, a computational graph model, and the XGBoost algorithm, SCGMEL introduces a hybridized methodology that deftly leverages the strengths of optimization and machine learning techniques, thereby substantially amplifying the approach's effectiveness in achieving optimal solutions.

At the forefront of innovation is SCGMEL's remarkable achievement in significantly enhancing computational efficiency. This advancement marks a major stride forward and is complemented by the pioneering application of XGBoost in predicting the optimal number of controllers required. In contrast to established algorithms such as ANSGA-III, NSGA-II, and MOPSO, the SCGMEL approach excels in terms of total CPU usage, including execution time and average CPU usage. Furthermore, the predictive capacity introduced by SCGMEL emerges as a remarkable innovation, surpassing other regression models and contributing substantially to more informed decision-making and improved accuracy in resource allocation. In summary, the SCGMEL represents a paradigm shift in tackling the challenges of the Controller Placement Problem within SD-WAN. Through the strategic fusion of innovative techniques, SCGMEL not only overcomes existing obstacles but also propels the field forward with enhanced efficiency, predictive capabilities, and optimal solutions.

The primary contributions of this study are summarized as follows:

- This study is the first to use a stochastic computational graph and an ensemble learning method to address the issue of CP in SD-WAN.
- The study optimizes controller location in the face of several competing objectives with the use of stochastic gradient descent with decay rate and momentum, as well as computational graphs.
- The study adopts a distributed XGBoost for the prediction of the ideal number of controllers in a software-defined wide-area network.

The following is an overview of the sections in this manuscript. Section 2 reviews the related works. The Section 3 explains the problem definition and objective functions used in optimizing the controller location in SD-WAN. Section 4 describes the proposed stochastic computational graph with an ensemble learning approach used to address the CPP in SD-WAN. In a similar fashion, the experiment, the result, its discussion, the verification and validation of the proposed SCGMEL, as well as the general conclusion, are each separated into Sections 5, 6, 7 and 8, accordingly.

2. Related works

The controller placement problem in software-defined networks (SDNs) has been a prominent focus of contemporary research endeavors. Researchers have presented various methodologies and algorithms to address this challenge, each contributing unique perspectives and innovations. Each approach offers distinct contributions, ranging from multi-objective optimization to machine learning and reinforcement learning techniques. While these methods may exhibit scalability and computational challenges (Multi-Objective Optimisation techniques), they collectively lay the foundation for innovative solutions that aim to enhance SDN controller placement and optimize network performance. This study's proposed stochastic computational graph approach, coupled with an ensemble learning model, holds promise as a novel solution to predict the optimal number and position of controllers in SD-WAN networks. This literature review provides an insightful overview of the key approaches while highlighting their strengths, limitations, and potential areas of application.

The fundamental importance of the Controller Placement Problem (CPP) is emphasized by Heller et al. (2012), who delve into the impact of controller placement on latency metrics and propose optimal solutions based on real-world models. The authors advocate for the significance of practical latency requirements, suggesting that many network architectures require only a single controller to fulfill these criteria.

Expanding upon the groundwork of Heller et al. (2012), Yao et al. (2014) introduce the Capacitated Controller Placement Problem (CCPP), considering controller load in addition to latency. The authors argue that controller load is a vital consideration and present a method to reduce the overall number of controllers required. Building on previous work, Hock et al. (2014, 2013) introduce the Pareto Optimal Controller Placement framework, which extends the analysis to include inter-controller latency, resilience, and load balancing. These authors leverage Pareto Simulated Annealing (PSA) to optimize multiple objectives simultaneously. However, the method is computationally expensive and requires a predetermined number of controllers, presenting potential limitations.

Multi-Objective Evolutionary Algorithms (MOEAs) have gained prominence in addressing the CPP. The seminal NSGA-II by Deb (2011) is an early example, enhanced by NSGA-III (Seada and Deb, 2014). Ahmadi and Khorramizadeh (2018) adapt NSGA to resolve multi-objective CPP issues in large networks, incorporating greedy heuristic and intelligent mechanisms. However, this method is computationally expensive and faces scalability challenges. Xu et al. (2022) in their study introduced a multi-controller load balancing model with a focus on optimizing three key objectives. The design aimed to achieve controller load balancing, reduce communication latency between the control plane and the forwarding plane, and minimize switch migration costs. The approach incorporated dynamic switch migration to distribute load across controllers efficiently, leading to reduced network latency and migration costs. Notably, the authors employed the Genetic Algorithm for Improved Multi-Objective Optimization to handle the challenges posed by fixed controller-switch mappings. While this approach showcases promise in optimizing network performance, scalability and computational efficiency could be potential concerns. Additionally, the algorithm's ability to predict the optimal number of controllers required for an SD-WAN network and to learn the heuristics of combinatorial optimization tasks remains to be determined. Further testing is needed to confirm its practical applicability.

Aravind et al. (2022) proposed the application of a simulated annealing algorithm to address the controller placement problem. In contrast with mixed-integer linear programming, the authors aimed to reduce execution time while optimizing controller placement. Their approach considers various conflicting objectives, including average and worst-case delays under controller failure and inter-controller delays. Notably, the study focuses on optimizing these objectives without considering switch-to-controller latency, a key factor in SDN controller optimization. While the simulated annealing approach accelerates optimal controller placement compared to mixed-integer linear programming, it may face challenges in efficiently addressing diverse solutions when multiple conflicting objectives exceed three. Furthermore, it might

encounter computational inefficiency, limitations in predicting the optimal number of controllers, and an inability to learn the heuristics of combinatorial optimization problems. Hemagowri and Selvan (2023) introduced a hybrid evolutionary algorithm, combining the artificial fish algorithm and chaotic Gaussian map, to optimize controller placement in SDN. Their approach strives to concurrently optimize controller placement based on various conflicting metrics, such as controller load, bandwidth, fault tolerance, data transmission rate, and propagation latency. To enhance solution diversity, a chaotic map extraction based on a Gaussian distribution was employed. While this approach exhibits efficiency in optimizing controller placement, challenges might arise in attaining diverse solutions for artificial fish within scenarios involving more than three conflicting objectives. Moreover, the inclusion of heuristic strategies during the training process might lead to computational intensity. Furthermore, the algorithm lacks the ability to learn the heuristics of combinatorial optimization problems specific to SD-WAN controller placement.

Kazemian and Mirabi (2022) addressed the controller placement challenge as a multi-objective optimization problem and introduced the Antlion optimization algorithm. Their study considers three objective functions: switch-to-controller delay, controller–controller delay, and separate connectivity links between nodes and controllers. The Antlion optimization procedure is a population-based evolutionary technique known for effectively optimizing conflicting objectives in SDN controller placement. However, challenges may arise when simultaneously optimizing more than three objectives, and the algorithm lacks the ability to learn the heuristics of combinatorial optimization problems. Compared to the stochastic computational graph approach proposed in this study, the Antlion optimization algorithm is known to be computationally expensive due to the various heuristic strategies involved in the training process.

Bagha et al. (2022) proposed a seagull optimization algorithm for controller placement, coupled with fuzzy C-Means clustering to partition the network. This approach aimed to optimize several objective functions, including end-to-end delay, inter-controller delay, node-to-controller delay, and controller load. The C-Means clustering algorithm was used for grouping controllers in close proximity to reduce overall network performance. However, uncertainties surround the practicality of this approach, particularly regarding fuzzy C-Means' application in partitioning networks with conflicting objectives. Additionally, population-based metaheuristics might face scalability issues in addressing multiple conflicting objectives. Moreover, this approach lacks the capability to learn the heuristics of combinatorial optimization, which could impact execution time.

Qaffas et al. (2023) proposed the adaptive group-based cuckoo optimization algorithm to address the controller placement challenge as a multi-objective optimization. Their study focused on optimizing an SDN-enabled wireless sensor network, considering reliability constraints, timing constraints, and controller installation costs. While the approach demonstrates the ability to find near-optimal solutions efficiently based on the considered objective functions, scalability concerns and limitations in predicting the required number of controllers remain. Additionally, the group-based approach lacks the ability to learn the heuristics of combinatorial optimization problems. In comparison to the stochastic computational graph approach proposed in this study, the Adaptive group-based cuckoo optimization algorithm may be computationally intensive due to the involvement of heuristic strategies during the training process.

Radam et al. (2022) propose a hybrid approach using harmony search and PSO algorithms for multi-controller-based SDN optimization. This technique optimizes multiple metrics while facing computational inefficiency challenges. Alouache et al. (2022) employ a Multi-Objective Genetic Algorithm combined with machine learning to optimize controller placement for Vehicular networks and 5G. While capable of calculating placements, this approach may suffer from scalability and computational inefficiency. Chen et al. (2022b) and Xiang et al. (2022) harness reinforcement learning techniques to optimize routing in SDNs and introduce optimal controller placements. These approaches exhibit potential for addressing the CPP, considering real-time network conditions. Yazdinejad et al. (2021) propose an SDN controller architecture for drone management, utilizing machine learning techniques. While not explicitly related to CPP, it showcases the integration of SDN and machine learning for optimized network performance. Wu et al. (2020) introduce a dynamic reinforcement learning-based approach to CPP, considering load balancing, flow fluctuations, and latency reduction. This technique exploits historical data and dynamic learning to achieve optimal controller placement. Chen et al. (2022a) present an ensemble and message-passing neural network-based deep reinforcement learning model for optical network routing optimization, offering robust decision-making capabilities. Thalapala et al. (2022) introduce a unique approach based on the wisdom of artificial crowds, achieving optimal controller placement while minimizing the number of controllers. Similarly, the cuckoo search algorithm, introduced by Sapkota et al. (2022), presents a population-based metaheuristic optimization technique for SD-WAN controller placement. The approach optimizes conflicting objectives like inter-controller delay, load balancing, and switch-controller delay. While efficient, scalability and the ability to learn heuristics may be limited.

Lastly, the authors in Adekoya and Aneiba (2022) introduced ANSGA-III as a solution for solving the CP challenge within SD-WAN. ANSGA-III aimed to optimize multiple competing objectives concurrently, addressing the complexities of SD-WAN controller placement. Despite its scalability, capable of handling more than three objectives, ANSGA-III was observed to be computationally demanding, setting it apart from the machine learning approach proposed in this study. Notably, ANSGA-III lacked the ability to acquire the heuristics essential for solving intricate combinatorial optimization tasks like SD-WAN CP. An additional limitation was the pre-determined determination of the number of controllers, potentially leading to either inflated installation costs or compromised network performance.

Consequently, a pressing necessity emerges for the development of a novel approach that blends computational efficiency and intelligence to effectively address the challenges of CPP within the realm of SD-WAN. This research endeavor seeks to fulfill this demand by harnessing the stochastic computational graph approach in tandem with ensemble learning, resulting in a well-rounded solution capable of predicting optimal controller numbers and locations while efficiently navigating the intricacies of SD-WAN controller placement.

3. Problem definition

In this research, the CPP in SD-WAN was analyzed. The approach in this research offered the ideal placement of controllers about SD-WAN architecture to satisfy multiple network criteria all at once. While switch-to-controller latency is the most important factor in a CPP scenario, other competing priorities must be considered. Among these goals are load balancing, average node-to-controller latency, maximum controller failure, maximum controller-to-controller latency, and controller–controller latency.

Based on the findings of this research, a controller placement method was developed as an unbounded, many-objective CPP. Eqs. (1) through (6) are where the objective functions of this research are presented to the reader. The SD-WAN is built in the form of an undirected graph represented by the equation $G=(\Omega, E)$, in which Ω represents the collection of switches and E represents the links that exist between the switches. In the meantime, to compute placement, a distance matrix denoted by the letter D contains information on the quickest routes between each pair of switches. The delay between switches α and node β is represented as $d_{\alpha\beta}$. Take note that the authors of this study arrived at their foundation for normalization by dividing the delay measured in D by the diameter of the corresponding graph.

The solution area is bound to a set of $\binom{m}{n}$ positions to get the desired result, which is specified by the number of controllers wanted.

According to the findings of this study, placement is referred to as a σ-element, where σ is a subset of Ω. The σ-subset of Ω, which is a space that contains all of the possible options, serves as the solution area for the CPP. Given a topology with 21 nodes and a specified number of controllers so that σ=7, for example, the set $\chi = 3, 4, 7, 9, 15, 16, 19\}$ signifies controller locations ($|PL| = 7$. This is an illustrated example. According to the assumption, the seven controllers need to be positioned at nodes 3, 4, 7, 9, 15, 16, and 33. It is important to keep in mind that moving the members about in any subset will not result in the creation of any new combination. Consequently, the overall number of possible locations in this network is $\binom{21}{7}$, given the circumstances of this particular instance. It is expected that a set of objectives known as $\{\beta_1, \beta_2, \ldots, \beta_m\}$ is to be minimized. When all other alternatives for possible placement have been exhausted, λ in the solution area, then solution χ is the optimal solution. In other words, $\forall_\alpha\ \beta_\alpha(lambda) \leq \beta_\alpha(\chi)$ and $\beta_\alpha(\lambda) < \beta_\alpha(\chi)$ no less than one index α. To address the controller placement problem with the SCGMEL, it is necessary to ensure that optimal solutions for the entire solution area reduce as the model progresses towards the optimal solution, demonstrating that the SCGMEL is capable of minimizing losses and reaching an optimal solution in less time.

3.1. Objective functions

In this section of the study, a comprehensive overview of the objectives under scrutiny is provided. For a more in-depth understanding of these objectives, readers are directed to refer to the study conducted by Lange et al. (2015). The process of controller placement involves addressing multiple conflicting objectives within a given configuration of controllers. These objectives encompass various performance goals that require careful consideration. The evaluation of the controller's worst-case and average switch-to-controller latency constitutes the first two performance metrics. In essence, these metrics capture the connection established between a switch and a controller. The determination of worst-case switch-to-controller latency, as outlined in Eq. (1), entails assessing the maximum latency observed across all switch-controller pairs within a particular member placement $PL \in 2^{\Omega}$ where Ω is the set of potential placement. Conversely, Eq. (2) computes the average switch-to-controller latency by considering the cumulative latency values between switches and controllers for a given distance matrix D, associated with the placement.

These calculations serve as pivotal indicators of network performance and connectivity efficiency, shedding light on the quality of interaction between switches and controllers within the placement configuration. The examination of worst-case and average latencies in the context of controller placement facilitates a comprehensive understanding of network responsiveness and latency management, crucial for optimizing the overall functioning of Software-Defined Networking (SDN) systems.

$$\eta^{Lat\_max\_N2C}(PL) = \max_{\omega \in \Omega} \min_{pl \in PL} d_{\omega,pl}, \tag{1}$$

$$\eta^{Lat\_avg\_N2C}(PL) = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \min_{pl \in PL} d_{\omega,pl}. \tag{2}$$

The real-world significance of these metrics becomes palpable when the tangible implications are put into consideration. Imagine a bustling network environment, where switches communicate with controllers to facilitate seamless data flow. Eqs. (1) and (2) encapsulate these real-world concerns by mathematically quantifying the worst-case and average latencies, respectively. These equations unravel the intricate relationship between placement configurations, distance matrices, and latency dynamics.

The switch-to-controller latency objective function contributes to a form of dynamic placement, even during the initial deployment of controllers. While the physical positions of the controllers may remain static, optimizing switch-to-controller latency ensures efficient and responsive communication paths between switches and controllers. In this context, "dynamic placement" does not necessarily mean physically moving controllers, but rather optimizing the network's responsiveness and adaptability. By minimizing switch-to-controller latency, the network is better equipped to handle varying traffic patterns, ensuring that data can quickly reach the appropriate controller for processing and decision-making. This responsiveness is essential for dynamic network management and efficient resource utilization. .

In the vast expanse of a large-scale network, the deployment of multiple controllers brings forth a pivotal need for seamless interaction and cohesive information exchange. The harmony between these controllers becomes paramount, as they work in concert to steer network operations. Amidst this intricate dance, a crucial factor emerges—inter-controller latency. It is a metric of paramount importance, one that holds the key to fostering efficient controller collaborations and thereby unleashing the network's true potential. When contemplating the strategic placement of controllers, the consideration of inter-controller latency becomes non-negotiable. The driving force behind this imperative lie in the quest to minimize latency, to ensure that the dialogue between controllers is swift, responsive, and conducive to network harmony. This is not a mere academic abstraction; it mirrors the real-world dynamics where controllers, much like the conductors of a symphony, must synchronize their actions seamlessly. Just as the formulation of Eqs. (1) and (2) serves a specific purpose, the expressions represented by Eqs. (3) and (4) fulfill a comparable role, albeit with a distinct focus on assessing inter-controller latency. Parallel to the treatment of switch-to-controller latency, the equations for inter-controller latency intricately account for both the utmost and average inter-controller latency possibilities. These performance metrics hold significant sway over the cooperative interactions among controllers, thereby necessitating a comprehensive investigation within the domain of CPP.

$$\eta^{Lat\_max\_N2C}(PL) = \max_{pl_1, pl_2 \in PL} d_{pl_1,pl_2}, \tag{3}$$

$$\eta^{Lat\_avg\_N2C}(PL) = \frac{1}{\binom{|PL|}{2}} \sum_{pl_1, pl_2 \in PL} d_{pl_1,pl_2}. \tag{4}$$

In the pursuit of minimizing latency and optimizing data transmission pathways, it is imperative to also address the crucial aspect of controller load balancing, which is vital for maintaining the efficient functioning of a network. While the other performance metrics considered in this study were designed for minimization, a distinct imbalance metric is introduced in this context instead of a balancing measure. This novel approach aims to achieve a well-distributed load across the controllers, promoting operational uniformity (Lange et al., 2015). While the physical positions of the controllers themselves might not change dynamically, the allocation of switches to controllers is being optimized in a way that ensures a balanced distribution of network load. This load-balancing process, even in the context of fixed controller positions, can be thought of as a dynamic optimization process because it aims to adapt and optimize the network's behavior over time and changing conditions. In the realm of dynamic controller placement, the number of devices allocated to a specific controller is denoted by the variable m_{pl}, where pl signifies a particular placement configuration. Concomitantly, m_{pl} represents the count of devices associated with the placement pl. To quantitatively capture load distribution disparities, the imbalance metric is formally defined by Eq. (5) (Lange et al., 2015). This equation effectively measures the difference in m_{pl} values between the two controllers, specifically the one with the lowest allocation and the one with the highest allocation of nodes. The introduction of this imbalance metric contributes to the dynamic placement of controllers, ensuring load-balancing objectives are dynamically incorporated into the overall optimization process.

$$\eta^{imbalance}(PL) = \max_{pl \in PL} m_{pl} - \min_{pl \in PL} m_{pl}. \tag{5}$$

Furthermore, an investigation into the possibility of maintaining resilient network operation in the event of controller failures was conducted as an essential objective within the optimization process of SD-WAN controller placement.

Consider a set denoted as $CL = 2^P \setminus \{\emptyset\}$ encompassing all potential controller positions remaining viable following the hypothetical failure of up to $(n-1)$ controllers. The average node-to-controller latency under various failure scenarios is theoretically modeled by Eq. (6).

$$\eta^{avg\_N2C}(PL) = \frac{1}{|CL|} \sum_{PL \in CL} \left(\frac{1}{|\Omega|} \sum_{\omega \in \Omega} \left(\min_{pl \in PL} d_{\omega,pl} \right) \right). \tag{6}$$

The consideration of resilient operation in the face of controller failures contributes to the dynamic placement of controllers. Ensuring resilience against controller failures is a crucial aspect of network design and optimization, especially in dynamic and rapidly changing environments like SD-WAN. By incorporating resilience as an objective function, the controller placement algorithm seeks to find positions that not only optimize other performance metrics but also ensure that the network remains operational even in the presence of failures. This contributes to the dynamic nature of controller placement as it necessitates adaptability and flexibility in the placement strategy to account for potential failures and maintain network stability. In this context, "dynamic placement" does not necessarily refer to physically moving controllers, but rather to the ability of the network to dynamically adjust its behavior and configuration to ensure continued operation and robustness in the face of failures. The resilience objective ensures that the network is able to maintain effective communication and operation even if certain controllers fail, which aligns with the concept of dynamic adaptation.

4. Proposed stochastic computational graph with ensemble learning model for SD-WAN controller placement

In this section, the research introduces the proposed solution, a stochastic computational graph with an ensemble learning model, designed to address the CPP in the context of SD-WAN. The CPP holds paramount importance, particularly within expansive enterprise networks like SD-WAN, where determining the optimal number of controllers to deploy and their strategic placement is a pivotal concern, complicated by competing objectives.

Past successes in solving controller placement issues in SD-WAN have been achieved through various metaheuristic algorithms, including MOPSO, ANSGA-III, and NSGA-II. However, these algorithms often encounter scalability challenges, particularly when the number of objectives exceeds three, and are associated with high computational demands. Notably, these solutions might lack the required level of intelligence to predict the optimal number of controllers needed for efficient SD-WAN deployment. To overcome these limitations, the proposed approach leverages a stochastic computational graph coupled with an ensemble learning technique, specifically XGBoost. This novel strategy aims to address the issues of scalability, intelligence, and computational complexity inherent in the aforementioned metaheuristic algorithms.

The proposed approach encompasses several critical components and algorithms, including stochastic gradient descent, an adaptive approach for objective function minimization, and the utilization of the PyTorch machine learning framework. PyTorch streamlines gradient computations and enables the construction of dataflow graphs, representing how data traverses the graph in multi-dimensional arrays. This not only automates gradient computations but also facilitates the translation of problem definitions into a format that enhances the computational graph model's effectiveness. The computational graph itself is employed to represent mathematical functions using graph theory, encapsulating the essence of various objective functions.

A notable feature of the solution is the integration of the XGBoost model, an ensemble learning technique, to predict the required number of controllers for an SD-WAN deployment. This predictive capability is invaluable in achieving an intelligent and well-informed controller placement strategy. The choice of XGBoost as an ensemble learning method is rooted in its proven ability to enhance predictive accuracy and model performance. XGBoost, as a boosting algorithm, falls within the category of ensemble methods, where the focus is on improving the capabilities of a single base model through iterative refinement. XGBoost achieves its ensemble effect through the construction of an ensemble model comprised of multiple decision trees. Unlike **ensemble of models** which involve combining multiple independently trained models, the ensemble model in this context pertains to the collaborative operation of multiple decision trees within XGBoost. It is important to clarify that in this research, the term **"ensemble model"** refers specifically to the collective behavior of the individual decision trees orchestrated by XGBoost. The emphasis is on iteratively enhancing the performance of a single model through sequential learning and boosting, rather than combining distinct independently trained models.

In summary, the proposed stochastic computational graph with an ensemble learning model offers a promising avenue for solving the complex challenges associated with SD-WAN controller placement. By leveraging the strengths of each component and algorithm, the approach strives to enhance scalability, intelligence, and computational efficiency while providing valuable insights into optimal controller placement for dynamic and efficient SD-WAN network operation.

4.0.1. Proposed stochastic computational graph algorithm description

The algorithm outlined in Algorithm 1 begins by requiring the input of the zoo topology dataset (Knight et al., 2011), as well as a data loader for loading tensors (representing controller locations) in batches. Additionally, the learning rate (a parameter influencing optimization step size) and the number of epochs (iterations) are specified. The algorithm then loads essential libraries for calculation (Line 2), such as Data Loader and PyTorch. The dataset is loaded (Line 3) and converted into tensor format (Line 4). The Haversine function is applied to compute distances between locations (Line 5), and the study's objective functions are computed using a computational graph technique (Line 6).

The algorithm generates a random initial controller location using the computational graph (Line 7), preserving a copy for later comparison (Line 8). A small amount of noise is introduced to the controller location in order to facilitate the computation of gradients (Line 9), and the instruction to track the operations of the tensor was sent to PyTorch (Line 10). A list is initialized to store losses, validation losses, and total losses (Line 11), with the final validation loss computed after all epochs (Line 12).

The initial losses are recorded as the best loss (Line 13), and the controller location is marked as the final location. The algorithm enters a loop (Line 14) iterating across sequences in the model. Through the data loader (Line 15), each tensor representing the controller location (Line 7) is processed. Objective functions are computed through the computational graph, and the computational graph enables the use of automatic differentiation to compute the gradients of the objective function, which can be used to update the parameters (controller locations) of the optimization algorithm. (Lines 17–18).

The controller location is updated based on learning rate and gradients (Lines 19–20), and all objectives are summed to form a single loss (Line 22). Position tensor losses and controller location losses are added to the training losses (Lines 23–24). Validation losses are accumulated and saved (Line 25), and if the current losses are lower than prior losses, the best loss and final location are updated (Lines 26–33).

A callback rate prevents overfitting (Lines 34–35), and print messages are displayed every four epochs. Line 36 executes after 10 epochs, checking for improvement and terminating algorithms if no progress is made. The best score is returned at an early stop (Line 42), along with optimal controller placement sites (Line 43). This algorithm optimizes SD-WAN controller placement, addressing conflicting objectives and leveraging an ensemble learning model within a stochastic computational graph framework.

The below explanation substantiates the statement in Line 22 of the algorithm that all objectives are summed to form a single loss

In the context of optimizing SD-WAN controller placement with six distinct objectives, it is crucial to differentiate between differentiable and non-differentiable objectives for a comprehensive evaluation of the optimization process and the eventual outcome. Among these objectives, four are differentiable, including average and worst-case switch-controller latency, as well as average and worst-case controller–controller latency. These differentiable objectives are amenable to a computational graph approach and can be optimized using gradient-based techniques, such as stochastic gradient descent (SGD), as demonstrated in this study.

On the other hand, the remaining two objectives, load balancing, and average controller resilience, are non-differentiable metrics. Unlike differentiable objectives, non-differentiable metrics cannot be directly optimized using gradient information, posing a challenge for standard gradient-based methods like SGD. In response to this challenge, a weighted approach is employed to transform non-differentiable objectives into scalar metrics, thereby enabling their inclusion within the optimization process.

In this study, each objective is assigned a weight to indicate its relative significance. This weighted approach facilitates the amalgamation of all six objectives into a unified optimization problem. By merging the differentiable objectives, which are amenable to gradient calculation, with the non-differentiable metrics, treated as scalar values with weights, a holistic optimization problem is formulated. This comprehensive approach considers all objectives and aims to achieve the optimal controller placement for SD-WAN while accounting for diverse performance criteria.

The incorporation of weighted values for non-differentiable metrics, accompanied by the weighted-based approach, is instrumental in transforming the individual objectives into a cohesive scalar function. Consequently, this approach allows for trade-off balancing between different objectives, culminating in the synthesis of a singular weighted objective function that encapsulates all six objectives. The allocation of weights to each objective provides the algorithm with the flexibility to navigate trade-offs, ensuring an optimal controller placement that aligns with multifaceted performance criteria.

This integrated approach substantiates the statement in Line 22 of the algorithm, wherein all objectives are aggregated to form a single loss. By leveraging the weighted-based approach, this study achieves the harmonious integration of differentiable and non-differentiable objectives, thereby enhancing the efficacy of gradient-based optimization methods for differentiable objectives, while appreciating the significance of non-differentiable metrics. The combined weighted objective function encapsulates all objectives, endorsing a comprehensive and multi-criteria approach to SD-WAN controller placement optimization.

4.0.2. Flowchart description of the proposed stochastic computational graph algorithm

The flowchart detailing the algorithm 1 is presented here, providing a succinct overview to enhance comprehension of the proposed method, as visually depicted in 1. The depicted flowchart serves as a guide to grasp the algorithm's functionality, illustrating the key steps and interactions.

Input Definition: The first box encompasses the algorithm's inputs, mirroring the parameters highlighted in algorithm 1. These inputs include the dataset (with the zoo topology and other relevant information), as well as fundamental tuning parameters such as the learning rate, momentum, decay rate, number of epochs, data loader configuration, and early stopping criteria. Corresponding to the details outlined in algorithm 1, each parameter's significance is reiterated.

Data Preparation: In the second box, the dataset provided in coordinate format undergoes a transformation into the tensor format, a requisite step for seamless integration with the computational graph framework. The Haversine function is then employed to compute the geographical distance between distinct locations.

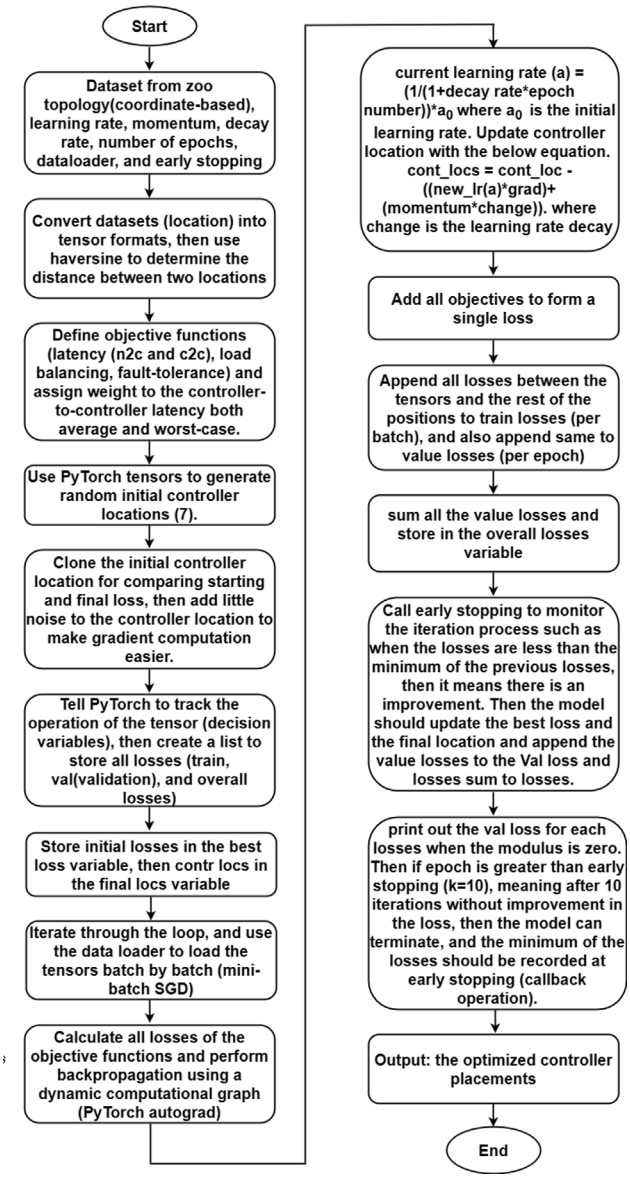


Fig. 1. The flowchart of the proposed Stochastic Computational Graph.

Objective Function Definition: The third box defines the set of objective functions central to the optimization of controller placement. These objectives encompass vital performance metrics, including maximum switch-to-controller latency, load balancing, average controller-to-controller latency, maximum controller-to-controller latency, average switch-to-controller latency, and maximum controller resilience. **Initial Controller Placement**: Box four illustrates the generation of the initial controller locations using PyTorch

tensors. To facilitate comparison between initial and final controller positions concerning loss functions, the initial positions are duplicated. A controlled amount of noise is introduced to the controller locations to facilitate gradient computation.

Loss Function Calculation: The fifth box delineates the process of calculating loss functions. PyTorch is instructed to monitor the operations involving decision variables (controller locations). Placeholder variables are defined to capture training loss, validation loss, and aggregate loss.

Iterative Optimization: The subsequent flowchart boxes mirror the iterative optimization process expounded in algorithm 1. As the description for algorithm 1 remains consistent for these subsequent

Algorithm 1 *Proposed Optimization Algorithm for SD − WAN Controller Placement using a Stochastic Computational Graph*

Require: *Dataset, Learning rate*($\emptyset\infty$), n_{epochs}, *dataloader*(dl), *early stopping, learning rate, decay rate, and momentum*
Load necessary libraries
Load datasets ($coordinate - based$)
Convert datasets into tensor formats
Define haversine distance function = $2(r)arcsin\left(\sqrt{sin^2\left(\frac{\beta_1-\beta_2}{2}\right)+cos(\beta_1)cos(\beta_2)sin^2\left(\frac{\alpha_1-\alpha_2}{2}\right)}\right)$
Compute objective functions using the computational Graph (CG) *approach*
Use CG to generate random initial controller location
Clone the initial controller location for later use
Add little noise to controller location to help in gradient computations
Tell PyTorch to track operation of the Tensor
Create list to store $train_{losses}$, val_{losses}, $losses_{sum}$
val_{losses}, = *calculate loss after all epoch is done*
$best_{loss} = initial_{losses}$, *and* $final_{locs} = cont_{locs}$
for *epochs in range*(n_{epochs}) **do**
  Run through batches using dl
  for pos_{tensor} *in dl* **do**
    Calculate loss2, loss, loss.backward ($PyTorchautograd$)
    $loss2 = first + second\ objective$, $loss = all\ objective$, *and loss.backward= calculate gradients* ($PyTorchautograd$)
    current learning rate $= 1 + decay\ rate * no_{epoch} * a_0$, *where* a_0 *is initial learning rate*
    update $cont_{locs} = cont_{locs} - new\ lr(a_1) * gradient + momentum * change$ $change$=$decay\ rate$
  end for
  Add all objective functions to form a single loss
  append $\left(all_{losses}(pos_{tensors}, cont_{locs})\right)$ *to* $train_{losses}$
  set $all_{losses}(pos_{tensors}, cont_{locs}))$ *to* val_{losses}
  sum val_{losses} *and store in losses*
  if $epoch > 0$ **then**
    if $losses < min(losses_{sum}$ **then**
      $final_{locs} = cont_{locs}.clone()$
      $best_{loss} = val_{loss}$
      append val_{loss} *to* val_{losses}
      append losses to $losses_{sum}$
    end if
  end if
  if $epoch$ mod $verbose$==0 **then**
    return each $val_{l}ossfor$ *each losses*
    if $epoch > early - stopping$ **then**
      if $min(losses_{sum}[-early - stopping :]) > min(losses_{sum}$ **then**
      end if
    end if
  end if
  return best score recorded at $early - stopping$
end for
Expected : *optimized controller placement positions*

steps, readers are encouraged to refer to the algorithm's explanation to avoid redundancy.

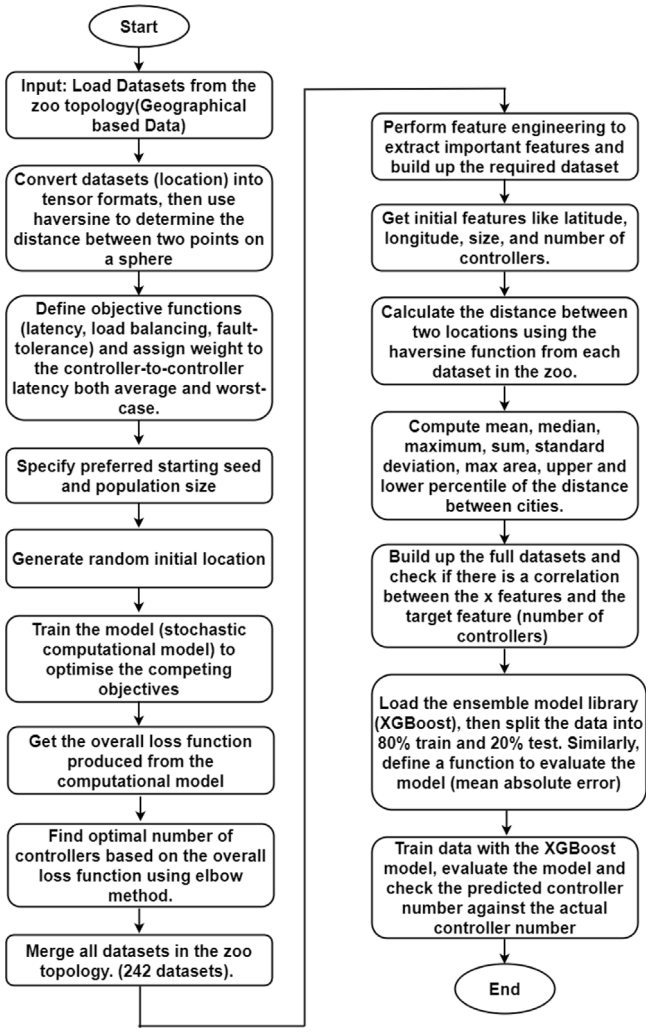


Fig. 2. The flowchart of the proposed Ensemble Learning method (XGBoost).

4.0.3. The flowchart of the proposed XGBoost model

By consolidating the flowchart description with that of algorithm 1, the aim is to eliminate repetition and ensure a seamless transition from textual to visual representation. This approach facilitates a deeper understanding of the proposed stochastic computational graph algorithm and its intricate details.

The proposed stochastic computational graph is detailed in Algorithm 1, while its corresponding flowchart is depicted in Fig. 1. In a parallel manner, the flowchart and algorithm encompassing the proposed XGBoost are presented. The descriptions of these components seamlessly follow the elucidation of the stochastic computational graph flowchart. This structured approach ensures a coherent and comprehensive understanding of the entire methodology.

4.0.4. The description of the proposed xgboost model flowchart

The flowchart, presented in Fig. 2, illustrates the XGBoost employed to determine the optimal number of controllers required for SD-WAN's optimal placement. This model plays a crucial role in enhancing the efficiency and effectiveness of controller deployment. The choice of XGBoost as an ensemble learning method is rooted in its proven ability to enhance predictive accuracy and model performance (Mamun et al., 2022). XGBoost, as a boosting algorithm, falls within the category of ensemble methods, where the focus is on improving the capabilities of a single base model through iterative refinement. XGBoost achieves its ensemble effect through the construction of an ensemble model

comprised of multiple decision trees. Unlike **ensemble of models** which involves combining multiple independently trained models, the ensemble model (XGBoost) in this context pertains to the collaborative operation of multiple decision trees within XGBoost (Montiel et al., 2020). The ensemble learning model, its methodology, and its application are detailed in the following discussion.

Input Loading and Preprocessing: The process begins with loading geographical datasets from the zoo topology (Knight et al., 2011), which consists of diverse WAN datasets from 242 service providers. These datasets are sourced from the zoo topology directory, a repository for real-world WAN datasets used for research purposes. The loaded datasets are then converted into a tensor format, and the haversine distance formula calculates distances between geographical locations.

Objective Function Definitions and Seed Value Specification: The third box establishes the objective functions used for calculating controller placement. Additionally, the population size and seed value are specified to ensure reproducibility and consistency in the results.

Random Initial Controller Location Generation: A random set of initial controller locations is generated to initiate the optimization process.

Stochastic Computational Graph Model for Controller Placement: In Box 6, the proposed stochastic computational graph model is utilized to optimize the decision variables based on the defined objective functions. This approach facilitates the identification of optimal controller placement, and the resulting overall loss plays a pivotal role in determining the required number of controllers.

Optimal Controller Number Estimation: Box eight employs the overall loss function obtained from the previous step to estimate the optimal number of controllers. This is achieved through the application of the elbow method, a technique that aids in identifying the inflection point where the overall loss starts to plateau. This point corresponds to the optimal number of controllers needed for SD-WAN deployment.

Feature Engineering and Dataset Preparation: Boxes nine to thirteen involve the preparation of data for the XGBoost model. Multiple steps are undertaken, including merging datasets from the zoo topology, performing feature engineering, and extracting essential features from the dataset.

Prediction of Optimal Controller Number with XGBoost: The subsequent sequence of boxes focuses on predicting the suitable number of controllers using the XGBoost model. This prediction is based on various dataset features and leverages feature engineering techniques.

Correlation Analysis and Model Preparation: In Box fifteen, correlation analysis is conducted between the features (X) and the target variable (number of controllers) to assess the model's potential performance. This step helps determine the suitability of the selected features for accurate prediction.

Training and Performance Evaluation with XGBoost: The subsequent stages encompass the training of the XGBoost model. The dataset is split into training (80%) and testing (20%) subsets, and the mean absolute error is utilized as a performance metric to evaluate the model's predictive accuracy.

Model Training Process and Optimal Controller Number Prediction: The final box encapsulates the training process of the XGBoost model, leading to the prediction of the required number of controllers. This prediction is based on the model's

learned patterns and insights from the feature-engineered dataset.

With the comprehensive understanding gained from the elucidation of flowcharts and their corresponding explanations, the research now introduces the proposed ensemble learning algorithm. As delineated in the depiction of the proposed XGBoost within the flowchart in Fig. 2, the identical elucidation provided for this flowchart is equally applicable to the description of the corresponding algorithm. The reader should please refer to the 4.0.3.

Algorithm 2 *Proposed Ensemble Learning Algorithm to Predict Optimal Controller Number in SD − WAN Controller Placement*

Require: *Required, coordinates based dataset*
Convert datasets into tensor formats
Define haversine distance function $= 2(r)arcsin\left(\sqrt{sin^2\left(\frac{\beta_1-\beta_2}{2}\right)+cos(\beta_1)cos(\beta_2)sin^2\left(\frac{\alpha_1-\alpha_2}{2}\right)}\right)$
Define objective functions using the computational graph approach
Specify seed and population size
Generate random initial controller location
Use Stochastic computational graph model to optimize decision variables with the given objectives
Obtain overall loss values from the model in previous step
Find optimal number of controller based on the overall loss using elbow method
Merge all 242 *datasets in the zoo topology*
Perform feature engineering to extract important features
Get initial features like latitude longitude number of controller, size of the datasets
Compute mean, median, maximum, sum, standard deviation, max area, upper and lower percentiles, of the distance between locations using feature engineering
Build up the datasets and check if there is correlation between the features and the target (number of controller)
Load the ensemble model XGBoost and split the data into X of 80% *and y of* 20%
Train the model and evaluate it with Mean absolute error
Expected : *the predicted number of controllers*

4.1. Description of the computational graph in the proposed solution

A computational graph is a potent tool that translates mathematical functions into a graph theory framework. It simplifies intricate mathematical relationships by organizing elements as nodes or edges in a graph (Evins, 2013). In a computational network, nodes can represent input values or functions that combine various values (Kim et al., 2022). Objective functions, pivotal in optimizing controllers, typically manifest mathematically, as showcased in Section 3. To leverage the potency of graph theory, this study embraces a computational graph approach to encapsulate objective functions. By mapping these functions onto a computational graph, the model efficiently interprets and executes them, thereby amplifying performance and operational fluency. Prominent machine learning frameworks, such as PyTorch and TensorFlow (Paszke et al., 2017) and De Rainville et al. (2012), heavily lean on constructing these computational graphs. These graphs facilitate back-propagation, vital for gradient calculation within a network. Backpropagation involves estimating the gradient of an input weight, which gauges loss change due to slight weight adjustment (Gao et al., 2020). To curtail overall loss and expedite training convergence, the weight undergoes iterative refinement using computed gradient, learning rate, and optimization parameters like momentum and learning decay rate.

The computational graph approach resonates with the multifaceted nature of SD-WAN controller placement, offering a unified framework for both differentiable and non-differentiable objectives. It seamlessly

integrates with machine learning models, embraces a weighted approach for conflicting metrics, and ensures scalability and efficiency for large-scale network optimization. This holistic approach empowers informed decisions, enabling optimal controller placement while accommodating diverse performance criteria.

The choice of the computational graph approach in this research is well-justified based on the following reasons: Optimization of Objective Functions: The computational graph approach is used to optimize the objective functions for SD-WAN controller placement. This approach is chosen because it allows us to systematically analyze and minimize multiple conflicting objectives, such as switch-to-controller latency, controller load balancing, and controller resilience. The computational graph provides a structured framework to represent and calculate these objectives, facilitating their simultaneous optimization.

Differentiable and Non-Differentiable Objectives: The proposed method involves both differentiable and non-differentiable objectives. Differentiable objectives, such as latency metrics, can be seamlessly integrated into the computational graph and optimized using gradient-based methods like stochastic gradient descent (SGD). On the other hand, non-differentiable objectives, like load balancing and controller resilience, pose challenges for traditional optimization algorithms. The computational graph allows us to handle both types of objectives within a unified framework.

Weighted Approach for Non-Differentiable Objectives: The computational graph's flexibility enables us to incorporate a weighted approach for non-differentiable objectives. By assigning appropriate weights to each objective, we can convert non-differentiable metrics into scalar values and include them in the optimization process. This ensures that the optimization process considers the trade-offs between different objectives, enhancing the overall quality of the controller placement.

Integration with Machine Learning Models: The computational graph seamlessly integrates with machine learning models, such as XGBoost. In the flowchart, we see how the computational graph's outputs, including the overall loss and controller locations, feed into the ensemble learning model. This integration allows us to harness the power of machine learning to predict the optimal number of controllers based on the computed losses and other features.

Scalability and Efficiency: The computational graph approach provides a scalable and efficient means of optimizing complex objective functions. This is particularly important in large-scale networks like SD-WAN, where numerous variables and objectives need to be considered. The graph structure optimizes computations and allows for parallel processing, leading to faster convergence and improved efficiency.

4.2. Description of the stochastic gradient descent in the proposed solution

Stochastic Gradient Descent emerges as a pivotal optimization algorithm extensively harnessed in the realm of machine learning (Yazan and Talu, 2017). Its seamless integration with back-propagation, particularly through frameworks like PyTorch, has solidified its status as a cornerstone for training neural networks. SGD stands as a strategic optimization approach uniquely tailored to tackle the intricate challenges posed by SD-WAN controller placement. The selection of SGD is firmly rooted in its remarkable compatibility with the computational graph model central to this study's methodology. As previously elucidated, the computational graph forms the backbone for optimizing objective functions, making SGD a natural fit for traversing this dynamic landscape. This symbiotic partnership between SGD and the computational graph manifests in a synergistic optimization process that navigates the intricate web of controller placement decisions.

The synergy begins with SGD's prowess in handling differentiable objectives, seamlessly aligning with the computational graph's capacity to manage such functions. This seamless alignment allows for a harmonious optimization process, refining placement decisions iteratively and culminating in well-informed outcomes. The iterative nature of SGD mirrors the step-by-step construction of the computational graph, forming a complementary relationship that propels the optimization journey forward. Crucially, the dynamic nature of SGD's learning rate brings adaptability to the forefront. This adaptability fine-tunes the step size to the unique contours of the optimization landscape. By dynamically adjusting the learning rate, SGD accelerates convergence speed and intricately refines controller placement—a critical aspect when striving to meet the diverse performance criteria intrinsic to SD-WAN. The symbiotic dance between SGD and the computational graph is further enriched by their parallel processing capabilities. As SGD's iterations unfold, the computational graph adapts concurrently, efficiently exploiting parallelism to hasten convergence. This seamless interaction caters to the demands of large-scale networks like SD-WAN, where numerous variables and objectives intertwine. In summation, the strategic adoption of SGD harmonizes with the dynamic computational graph model, culminating in an optimization strategy that adeptly leverages the graph's versatility. This union navigates the intricate network terrain with efficiency and precision, orchestrating an iterative ballet that converges towards optimal placement decisions, each step informed by the seamless interplay between SGD's adaptability and the computational graph's dynamic construction.

4.3. Learning rate

In Chee and Toulis (2018), the impact of the learning rate on the gradient descent algorithm is emphasized. The learning rate determines the scale of parameter updates as the algorithm progresses downhill. It has a dual role in influencing the speed of knowledge acquisition by the model and the reduction of the cost function. Machine learning involves two types of parameters: machine-learnable parameters adjusted autonomously by algorithms and hyper-parameters set by data scientists. The latter fine-tune the learning process to enhance predictive accuracy. Represented as α, determining the optimal learning rate is challenging. This research employs a time-decaying learning rate, making the learning process dynamic rather than static, which adapts to the problem's complexity over time. This approach ensures a flexible and precise optimization trajectory.

4.4. Stochastic gradient descent with momentum

Incorporating momentum into stochastic gradient descent, known as SGD with momentum (Liu et al., 2020), introduces a memory element that retains a portion of the prior vector update while computing the next one. This augmentation enhances the optimization process by helping SGD navigate past local minima and converge towards the global optimum. Combining momentum with SGD accelerates optimization and aids in avoiding the challenges of selecting an optimal learning rate, which can lead to getting stuck in local minima or slow convergence. In the context of the proposed solution's optimization phase, this investigation introduces historical information into parameter updates through momentum. The momentum component is influenced by gradients observed in previous iterations. By incorporating momentum, the proposed technique mitigates the impact of increased learning pace and ensures training consistency. It is crucial to retain a record of recent vector modifications to account for them in subsequent computations, contributing to the overall stability and effectiveness of the optimization process.

4.5. Learning rate decay

Learning rate decay, as discussed in You et al. (2019), is instrumental in guiding the model towards the global optimum and mitigating oscillations observed in traditional SGD without momentum and learning rate decay. The recommended SGD optimization model integrates

an adaptive learning rate decay strategy, which assesses the significance of prior updates in the optimization process. This approach gradually moderates the training pace, reducing manual intervention and parameter adjustments. The outcome is a reduction in oscillation and expedited convergence, underscoring the advantages of a moderated learning rate.

4.6. Execution time

The execution time of a computer program holds significant importance in assessing system performance (Stattelmann et al., 2014). It reflects the duration for executing instructions within algorithms or models. The goal is to minimize execution time, achieved by optimizing the number of steps needed to complete tasks. Tracking time intervals involves counting clock ticks between the start and end of events, yielding accurate measurements. In the realm of computer science and software engineering, execution refers to the implementation of program commands, with each code line representing a distinct task. Evaluation of non-functional aspects often necessitates estimating execution time. To this end, techniques fall into static and dynamic categories. Static methods predict task completion time through structural code analysis and hardware modeling. Dynamic approaches involve running the program for estimation, sometimes coupling static analysis for prediction. Notably, static timing analysis estimates worst-case execution time. This research employs dynamic timing analysis on the proposed and state-of-the-art optimization algorithms to estimate execution time. Run-time libraries play a role, and execution time involves interpreting instructions to execute tasks.

4.7. Average CPU usage

Average CPU Utilization: Average CPU usage quantifies the workload a CPU handles to complete assigned tasks (Domingues et al., 2005). Elevated CPU usage can detrimentally affect system performance, necessitating vigilant monitoring. The Python System and Process Utilities module, or psutil, provides insights into active processes and system utilization (CPU, memory, load). Primarily employed for system monitoring and process management, it gauges core load percentages for performance assessment. Evaluating hardware resource consumption during program execution is vital to gauge program effectiveness and its impact on resource utilization.

4.8. Total CPU usage

Total CPU Usage: Total CPU usage quantifies the entirety of processing power allocated to execute an algorithm (Rashid et al., 2018). The cumulative CPU percentage employed since the initiation of the procedure is denoted as the total CPU percent. It signifies the comprehensive CPU utilization by the process from its inception. In this study, the total CPU was computed by multiplying the execution time with the average CPU utilization. The execution time of the algorithm was measured using the time module (DateTime) in the experiment. The datetime.now() function was invoked before the script's commencement, while the end time was recorded using the same method just before the script's culmination. The time module serves diverse purposes in this experiment, encompassing timestamping at procedure initiation and conclusion, ultimately calculating the ''execution time'' by determining the temporal difference between the two timestamps.

4.9. Mean absolute error

Mean Absolute Error (MAE), as described in Qi et al. (2020), represents the average magnitude of discrepancies between actual and predicted values. Another interpretation characterizes it as the mean difference between observed data and predicted outcomes. By disregarding the sign of deviations, the potential for offsetting between opposing numerical values is mitigated. In the context of this research, the efficacy of a regression model is assessed through the MAE metric. A higher MAE, considerably distant from zero, indicates sub-optimal performance, while an MAE closer to or equal to zero signifies a nearly flawless model with minimal prediction errors. It is important to note that the interpretation of MAE's effectiveness depends on context. For instance, in this study, a lower value of MAE, approaching zero, is considered indicative of superior model performance.

5. Experimentation

For this experiment, a computer with an Intel Core i7-6820HQ processor running at 2.70 GHz, 1600 MHz DDR3 RAM with 64 GB, and a Microsoft Windows 10 Professional Edition operating system was used. For the purpose of the experiment, Python programming language was utilized. In order to build the code, Jupyter notebook version 6.3.0 was utilized. The code is accessible as open-source with the link provided at this web address https://tinyurl.com/2p95ad26. In this study, the automatic differentiation and computation of gradients were done with the help of the machine learning package known as PyTorch. TensorFlow PyTorch was used to create the decision variables, which were then changed to tensor format to make PyTorch easier to use. A tensor container was used to store the initial location of the controller. This location was chosen at random from the members of the dataset. In order to load the tensors in batches, a dataloader was employed. Multiple competing objectives for SD-WAN controller location were optimized using stochastic gradient descent. The number of controllers was predicted using XGBoost, a gradient-descent tree regression model.

The main purpose of this research is to determine how many SD-WAN controllers are required to build up the topology and where in the WAN topology to place them in order to achieve various goals. The WAN topology used in this study was gotten from the Internet Zoo topology (Knight et al., 2011). This study makes use of the entire datasets in the zoo topology, which consist of 242 datasets from different service providers, for the prediction of the number of controllers, while one dataset (BtEurope) was used for the placement of controllers, so that competing objectives may all be met. Feature engineering was performed on the available data to extract important features which help in building the data used in the number of controller predictions. The X features consist of features such as the longitude, latitude, size of the datasets, the mean distance, maximum distance, and standard deviation, the lower (25%) and upper (75%) percentile. Meanwhile, the y features which represent the target variable consist of the number of controllers. There are a total of six objectives to this research: latency from switch to controller on average, maximum latency from switch to the controller, latency from the controller to the controller on average, maximum latency from the controller to controller, resilience regarding controller failure, and load balancing (load imbalance). The settings were established using the following variables. The learning rate was 0.01, the total amount of objectives was 6, the number of tensors (controller location) was 7, the epoch numbers were 200, the early stopping (callback operation) was 10, and the seed value (reproducible) was 12. The training data was 80% while the test data was 20%.

6. Results and discussion

This section presents a comprehensive summary of the experimental outcomes. The study undertakes a comparative analysis, juxtaposing the results of the Stochastic Computational Graph with Ensemble Learning against those of ANSGA-III, NSGA-II, and MOPSO algorithms. The culmination of these analyses is visually depicted in Figures 3 to 19. The central objective of this research revolves around strategically positioning SD-WAN controllers within the WAN topology to attain diverse objectives while optimizing the number of controllers. To attain the insights elucidated in this study, six critical aspects of the SD-WAN

Fig. 3. Image showing the BtEurope Dataset.

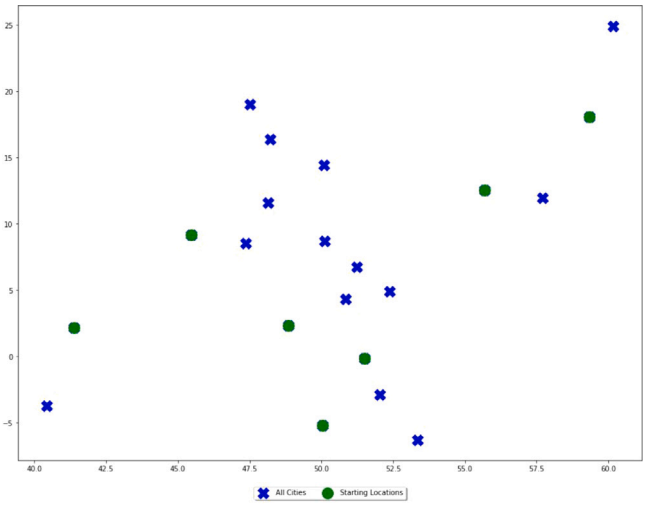


Fig. 4. Image showing the Starting Controller Location.

```
Our starting Cities: [ 7  9 10 20 13  8 14]
Our starting Locations: [[48.85342     2.34881   ]
 [41.3888      2.159     ]
 [50.05001    -5.19999   ]
 [59.332592   18.064909  ]
 [55.675953   12.565539  ]
 [45.464283    9.18952   ]
 [51.50854    -0.12573001]]
Initial maxN2c: 489.01 meanN2C: 441.27 maxICL: 1416.41 meanICL: 1452.65
Overall for initial locs: 4988.71
```

Fig. 5. Image showing the Final Controller Location.

controller placement quandary were scrutinized. These aspects encompassed vital performance indicators such as execution time, average CPU usage, average core load, total CPU usage, and regression model performance gauged by the mean absolute error. The pivotal attributes under consideration encompassed latency from average switch to controller, maximum switch-to-controller latency, average controller-to-controller latency, maximum controller-to-controller latency, controller resilience in the face of failures, and load balancing.

In the pursuit of equitable comparison, both the proposed algorithm and the bench-marked counterparts were subjected to identical data observations. Parameters under scrutiny encompassed initial and final controller placements, execution time, average CPU usage, average core load, and total CPU usage—with the latter computed as the product of execution time and average CPU utilization. Similarly, in a bid to facilitate comprehensive comparison and validation, all algorithms underwent meticulous amalgamation. The transition of initial and final controller positions was mapped onto corresponding location data frames, articulated in a coordinate-based format. All the algorithms were combined to enhance comparison and validation. The initial and final controller locations were mapped back to the corresponding location data frames (coordinates-based form), and the haversine distance function was used to calculate the distance between the locations.

The initial and final losses were determined based on the initial and final controller locations, utilizing the respective objective functions. The disparity in total loss between the initial and final states was then computed. This process was executed for all pertinent objective functions. Subsequently, the average of these losses across all objectives within each algorithm was computed for assessment. These procedures serve as a basis for evaluating the models' performance.

The subsequent sections delve into the analysis and interpretation of the experimental outcomes. To corroborate the presented results, readers are encouraged to refer to the provided link for validation: https://tinyurl.com/2p95ad26.

The visual representations encapsulated in Figs. 3, 4, and 5 provides an insightful depiction of the BTEurope dataset, the initial and final controller placements orchestrated by the stochastic computational graph model. Fig. 6 emerges as a pivotal cornerstone, unveiling the initial controller's spatial disposition, its corresponding coordinates, and the encompassing initial loss. This image assumes paramount significance as it serves as a barometer to gauge the model's efficacy post-controller optimization. From the contours of Fig. 6, a discernible initial controller overall loss of 4988.71 emerges, encapsulating the baseline performance metrics. In tandem, Fig. 7 occupies the spotlight, unfurling the overarching loss achieved by the proposed stochastic computational graph model without the inclusion of decay rate and momentum. Evidently, the model converges at epoch 209, culminating in an overall loss of 1580.04. This visualization underscores the model's propensity to optimize and refine its performance over iterations. Fig. 8 elegantly charts the trajectory of overall training and validation, juxtaposed against the total loss. This compelling graph unveils a noteworthy trend: as the number of epochs escalates, the proposed computational graph model diligently curbs the loss function. This phenomenon, whereby the loss steadily diminishes with escalating epochs, signifies the model's progressive quest for convergence. A pivotal facet here is the employment of a callback operation (early stopping), a strategic maneuver to preempt overfitting. This operation ensures that the model deftly halts when further training ceases to yield a reduction in loss, effectively attaining convergence and forestalling unnecessary iterations. Collectively, these graphical representations interweave to construct a nuanced narrative, underscoring the efficacy of the stochastic computational graph model and its intricate interplay with vital components such as decay rate, momentum, and early stopping mechanisms.

The visual representation encapsulated in Fig. 9 offers a profound insight into the overarching score and the pivotal epochs that culminate in model convergence. This outcome is a direct consequence of the strategic incorporation of decay rate and momentum into the model's architecture. Notably, the convergence point is distinctly marked at epoch 66, signifying a critical juncture in the optimization process. The crux of this analysis revolves around the profound impact of momentum and decay rate on the proposed stochastic computational model. It is manifestly evident that the integration of these dynamic elements imparts a remarkable advantage. Comparatively, when juxtaposed with

the stochastic computational model lacking momentum and decay rate, as exemplified in Figs. 8, 7, and 6, the proposed solution showcases a more efficient trajectory. Illustratively, Fig. 10 systematically dissects each individual objective function alongside the cumulative total loss, presenting a comprehensive panorama of the optimization landscape. Additionally, Fig. 11 emerges as a central pivot, spotlighting the culmination of the stochastic computational model embellished with momentum and decay rate. Evidently, the figure substantiates the model's achievement of an overall loss of 883.41, attained through a notably expeditious convergence process spanning a mere 66 epochs. In essence, the collective narrative woven through these visual representations underscores the indispensable role played by momentum and decay rate in elevating the efficiency and efficacy of the proposed stochastic computational model. This convergence of evidence imparts a robust rationale for their integration, amplifying the model's potency in arriving at optimal solutions swiftly and with heightened precision.

Fig. 12 emerges as a pivotal visual aid, unraveling the imperative facet of determining the requisite number of controllers for seamless SD-WAN deployment, with a specific focus on the BtEurope network scenario. The same methodology is seamlessly extended to encompass all datasets within the zoo topology (Knight et al., 2011), ensuring a comprehensive and consistent evaluation. The foundation for ascertaining the optimal number of controllers rests upon the bedrock of the overall loss function, as conspicuously depicted in the graph. Within this analytical framework, a noteworthy juncture, known as the point of inflection, materializes. This pivotal juncture is found at a count of 7 controllers. It is essential to engage in a judicious analysis of the trade-offs inherent in this context. Opting for 14 controllers, while potentially yielding enhanced network performance, could incur a substantial financial burden for service providers. Conversely, selecting a more frugal configuration, with 7 controllers, offers a prudent alternative that does not significantly compromise the overall loss metric. This delicate balance between performance and cost underscores the significance of a nuanced decision-making process. Therefore, the evidence presented substantiates the notion that opting for a configuration of 7 controllers emerges as a pragmatic and astute choice for service providers embarking on SD-WAN deployment. This prudent decision effectively navigates the trade-offs and encapsulates the essence of optimizing the network infrastructure for optimal performance while judiciously managing costs.

Fig. 13 serves as a comprehensive visual representation, shedding light on the execution time attributes of various models employed in the evaluation process. The proposed algorithm, characterized as the stochastic computational graph model, stands juxtaposed with three other models: ANSGA-III, NSGA-II, and MOPSO. Upon meticulous scrutiny of the graph, a discernible trend materializes: the proposed solution remarkably excels in terms of computational efficiency, as reflected in its notably lower execution time. This advantageous trait is particularly highlighted by the fact that the proposed solution outperforms the other compared algorithms by an impressive margin of almost 100%. This performance discrepancy underscores the prowess of the proposed algorithm in achieving faster optimization of controller location. The commendable computational efficiency and scalability exhibited by the proposed solution position it as a highly recommended choice for service providers contemplating the migration or deployment of SD-WAN architecture. The algorithm's ability to strike a harmonious balance between swift execution (lower execution time) and accommodating a multitude of objectives renders it an attractive and pragmatic option for SD-WAN planning endeavors.

Fig. 14 provides a clear depiction of the average CPU utilization exhibited by both the proposed algorithm and the three alternative models considered in the evaluation process. Similarly, akin to the execution time graph illustrated in Fig. 14, this graph portrays the salient observation that the proposed solution demonstrates the lowest CPU usage across the spectrum of algorithms studied.

```
Our starting Cities: [ 7  9 10 20 13  8 14]
Our starting Locations: [[48.85342     2.34881   ]
 [41.3888      2.159     ]
 [50.05001    -5.19999   ]
 [59.332592   18.064909  ]
 [55.675953   12.565539  ]
 [45.464283    9.18952   ]
 [51.50854    -0.12573001]]
Initial maxN2c: 489.01 meanN2C: 441.27 maxICL: 1416.41 meanICL: 1452.
Overall for initial locs: 4988.71
```

Fig. 6. Image showing the initial controller location, its coordinates, and the overall loss.

```
Training ended by early stopping, best score at epoch 209
Best scores at epoch 209 maxN2c: 247.27 meanN2C: 365.30 maxICL: 14.56 meanICL: 14.89 Overall: 1580.04
```

Fig. 7. Image showing the overall loss of the stochastic computational graph model without momentum and decay rate.

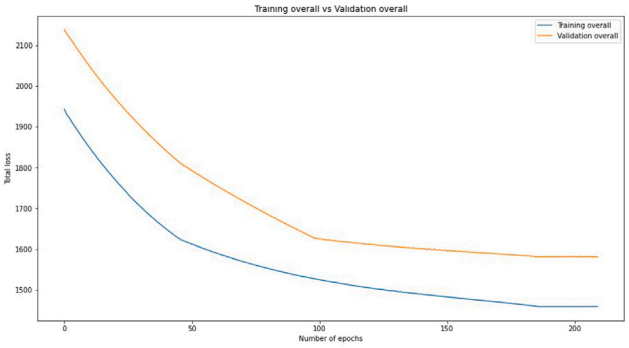


Fig. 8. Graph of overall training and validation of the stochastic computational model.

```
Training ended by early stopping, best score at epoch 66
Best scores at epoch 66 maxN2c: 193.08 meanN2C: 408.87 maxICL: 1630.59 meanICL: 1606.86 Overall: 883.41
```

Fig. 9. Image showing the overall loss of the stochastic computational graph with momentum and decay rate.

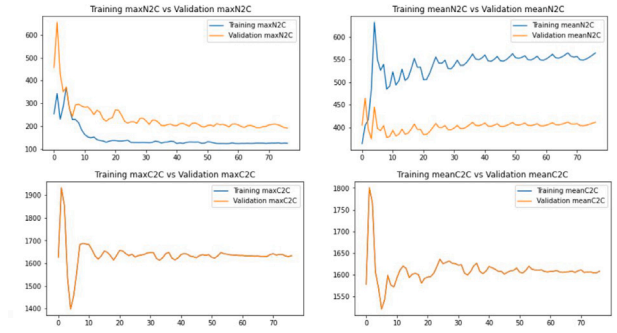


Fig. 10. Graph representing individual objective function overall loss of the stochastic computational model.

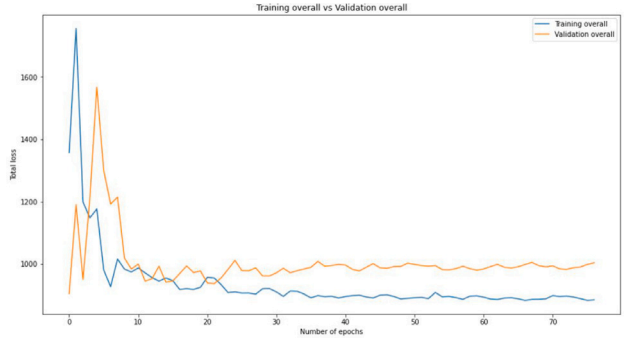


Fig. 11. Graph of overall loss of stochastic computational model with momentum and decay rate.

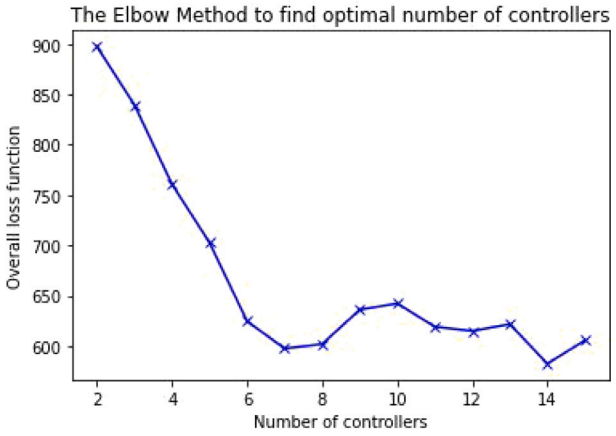


Fig. 12. Graph of the number of controller.

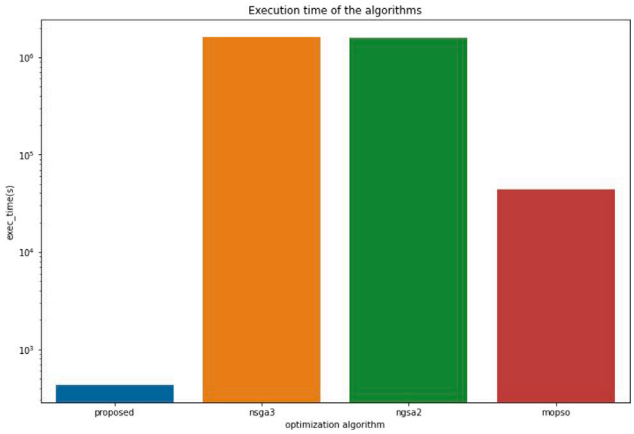


Fig. 13. Graph showing the execution time of the four Algorithms.

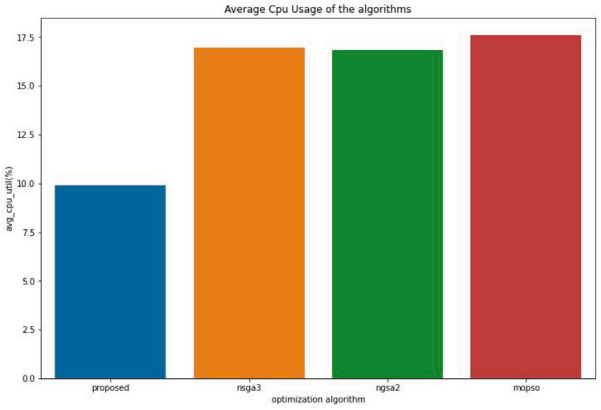


Fig. 14. Graph showing the average CPU usage of the four Algorithms.

Remarkably, the proposed SCGMEL solution distinguishes itself by showcasing superior performance in comparison to its counterparts. The magnitude of this performance improvement is vividly highlighted in the percentages of reduction achieved. Specifically, the SCGMEL solution outperforms ANSGA-III by a substantial reduction of 99.983%, NSGA-II by an equivalent reduction of 99.983%, and MOPSO by an impressive reduction of 99.946%. This substantial margin of performance enhancement positions the SCGMEL solution as the clear choice, thus warranting its strong recommendation. Furthermore, Fig. 15 elucidates the total CPU usage exhibited by the proposed algorithm alongside the three alternative models considered in the evaluation. The discernible pattern in this graph reaffirms the trend observed in other computational complexity metrics, including execution time and average CPU utilization. The proposed solution consistently outperforms its algorithmic counterparts in terms of computational demands.

In summation, the overarching performance assessment based on computational complexity unequivocally positions the proposed stochastic computational graph model as a formidable contender. Its superiority is pronounced not only in individual metrics but across a comprehensive range, making it the recommended choice for deployment in the context of SD-WAN.

Fig. 16 serves as a visual representation encapsulating the essence of all the optimization algorithms scrutinized in this study. The graph magnifies their respective performances in the context of minimizing losses, all conducted under identical experimental conditions and objective functions governing the optimization of controllers. Upon careful examination of the graph, a clear pattern emerges: the proposed solution stands out prominently by consistently achieving a greater reduction in losses compared to its algorithmic counterparts. This discernible trend underscores the efficacy of the proposed approach in efficiently minimizing losses, positioning it as a standout performer in the realm of optimization algorithms.

The inferential statistics for both the proposed XGBoost model and other compared regression models (KNN, random forest, and linear regression) are presented comprehensively in Table 1. Additionally, Fig. 17 visually presents the regression models employed for predicting the optimal number of controllers required for SD-WAN deployment. Notably, the proposed XGBoost model is subjected to a rigorous comparison with the aforementioned regression models. Evaluation of these models hinges on the mean absolute error metric, shedding light on their performance. Remarkably, the outcomes showcased in the graph distinctly highlight the superiority of the proposed solution. With an impressively low test mean absolute error of 1.855, coupled with an efficient prediction time of 0.004, the proposed regression model not only operates expeditiously but also exhibits unparalleled accuracy when contrasted with its machine learning regression counterparts. In a parallel vein, Fig. 18 delves into the performance analysis of the proposed regression model concerning the prediction of the number of controllers. This insightful graph accentuates the juxtaposition between randomly selected datasets from the zoo topology on the x-axis and the corresponding actual versus the predicted number of controllers on the y-axis. The graphical representation elegantly encapsulates the model's predictive prowess, providing a visual affirmation of its efficacy in approximating the optimal controller number. The graph clearly illustrates that the proposed regression model closely predicts the actual number of controllers across the majority of selected datasets. This noteworthy accuracy highlights the effectiveness of the model's predictive capabilities. As a practical implication, it is advisable for SD-WAN service providers and administrators to embrace the methodology outlined in this study when determining the appropriate number of controllers for their deployment. By implementing this approach, they can significantly mitigate control plane overhead, leading to a substantial enhancement in overall network performance. This not only streamlines the decision-making process but also contributes to the optimization of SD-WAN operations, ultimately translating into improved service quality and operational efficiency (see Fig. 19).

6.1. Statistical analysis of final output loss for proposed and benchmark algorithms

The objective of this analysis is to conduct a statistical assessment to determine whether the proposed controller placement algorithm indeed exhibits a significantly lower output loss when compared to the reviewed MOPSO, NSGA-II, and ANSGA-III algorithms. The significance

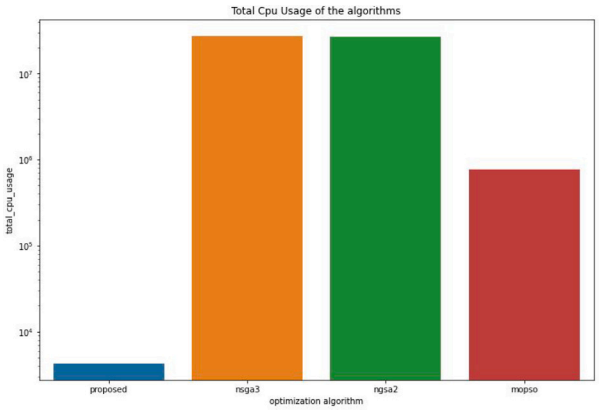


Fig. 15. Graph showing the total CPU usage of the four Algorithms.

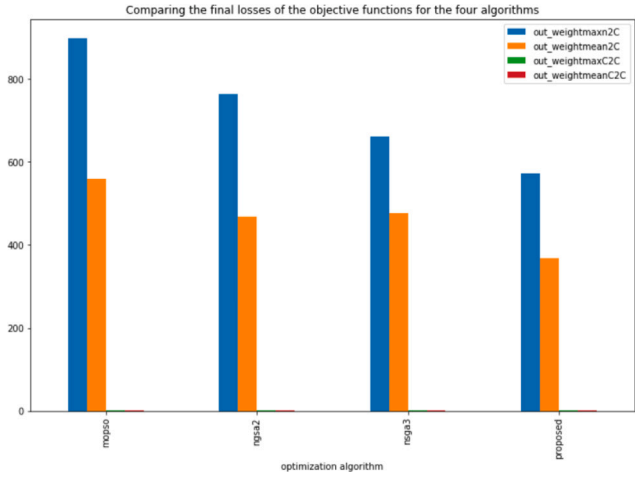


Fig. 16. Graph showing the final losses of the four Algorithms.

Table 1
Inferential statistics table of all the regression models.

	fit_time(secs)	pred_time(secs)	train_mae	test_mae	model
0	0.011894	0.037631	3.390244	3.829268	knn
1	0.004931	0.011087	4.436261	3.729883	lr
2	1.124537	0.218368	1.334114	1.883536	rf
3	0.061276	0.004411	0.089354	1.855751	xgb

of this analysis lies in its potential to identify the most recommended controller placement algorithm that demonstrates the least significant output loss within the SDN context. Prior to delving into the statistical analysis, this study will first establish the relevance of the proposed approach by comparing it to relevant and similar models (see Table 2). In comparison to existing models in the field, the reported approach presented in this research offers distinct advantages and improvements. While several optimization techniques, such as MOPSO and ANSGA-III, have been previously employed for SDN controller placement, the proposed approach introduces a comprehensive solution that addresses key limitations and provides superior performance (see Fig. 20).

MOPSO, NSGA-II, and ANSGA-III, although valuable, have shown challenges in scalability (except ANSGA-III), computational complexity, and lack the ability to predict the required number of controllers needed for SD-WAN deployment. In contrast, this research introduces a novel methodology: The stochastic Computational Graph Model with Ensemble Learning. These approaches are specifically tailored to overcome these challenges and optimize multiple conflicting objectives simultaneously (see Fig. 26).

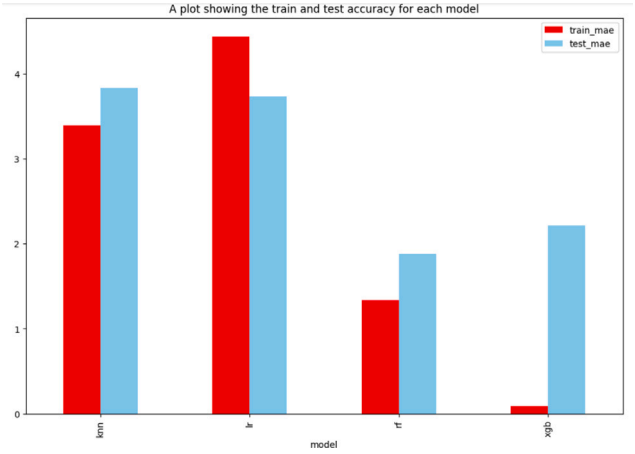


Fig. 17. Graph showing all the regression models for optimal number of controllers prediction.

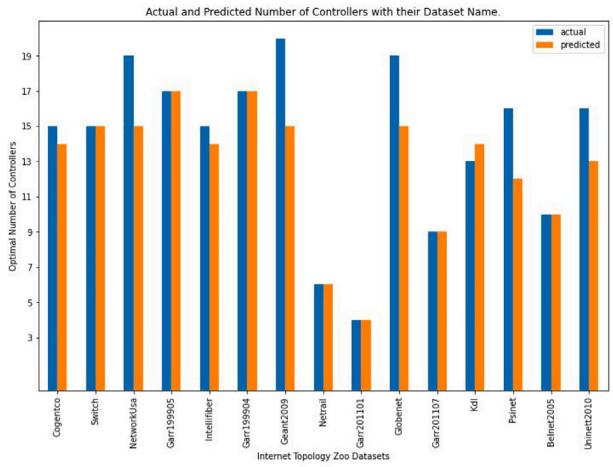


Fig. 18. Graph showing the proposed XGBoost regression model for the prediction of controller.

98	5	7	Getnet
30	15	17	BtLatinAmerica
25	20	13	Bics
16	10	10	Belnet2003
168	8	7	Restena
195	13	16	Uunet
97	15	15	Geant2012
194	11	12	UsCarrier
67	12	13	Garr199901
120	15	14	Intellifiber
154	10	10	Ntt
202	19	15	York
79	19	19	Garr201004
69	17	17	Garr199905
145	6	6	Netrail
55	17	15	Digex
45	15	14	Cogentco
84	9	9	Garr201104
146	19	15	NetworkUsa

Fig. 19. Graph showing the proposed XGBoost predicted and actual controller number.

```
The network final output loss means of the algorithms are:
 proposed     936.018623
mopso        1459.347972
nsga2        1221.599093
nsga3        1142.605371

The network final output loss variances of the algorithms
 proposed    25139.335528
mopso         3067.111557
nsga2        32561.492155
nsga3        33732.179519
```

Fig. 20. Final output loss means of the optimization algorithms.

Table 2
Sample datasets illustrating the final output loss values obtained from various optimization algorithms.

	proposed	mopso	nsga2	nsga3
0	716.016170	1493.835447	1318.060786	1205.792011
1	714.692806	1493.928038	1443.265407	1079.906553
2	1073.149020	1389.168394	844.720857	1024.933980
3	690.896103	1492.154173	773.148163	1135.833079
4	1013.050896	1389.585855	1157.688084	1220.144165
5	1013.075670	1540.748289	1396.240934	939.967940
6	1013.146655	1365.462906	1021.786994	1039.683992
7	999.735857	1438.179971	1220.144165	1275.039106
8	737.644693	1387.394529	1364.921533	1028.227812
9	865.387042	1389.075804	797.485422	1200.618064
10	1331.641651	1438.347339	1117.091896	773.188143

The relevance of the reported approach is its computational efficiency and intelligent nature. This was demonstrated in the study as revealed by the execution time, average CPU, and Total CPU usage performance metrics which are all used to assess the performance of the optimization algorithm. By using a computational graph approach, you can model the complex interactions between the different objectives, while using a machine learning framework such as PyTorch or TensorFlow can provide efficient computation and optimization. Stochastic gradient descent with learning rate decay and momentum can help to improve the convergence rate and the quality of the solutions found, while the weighted sum approach can help to find a solution that balances the trade-offs between the different objectives (see Fig. 22).

Contrary to this approach, evolutionary algorithms are more computationally complex than machine learning approaches, particularly if the search space is large and complex. Evolutionary algorithms involve iteratively generating and evaluating a population of candidate solutions using heuristics such as selection, crossover, and mutation, which can require a significant number of computational resources, especially when searching for an optimal solution in the presence of conflicting objectives. Similarly, the ensemble learning model (XGBoost) used in this research helps in the prediction of the optimal number of controllers to be deployed in the SD-WAN environment. In summary, the reported approach not only outperforms existing models in terms of execution time, computational efficiency, and scalability but also introduces a holistic solution by integrating an intelligent solution that provides the optimal number of controllers to be deployed. This clear distinction and innovative integration highlight the relevance and superiority of this research approach when compared to relevant similar models (see Figs. 23–25).

For clear conceptual justification, this work computed the final losses of the objective functions of each of the optimization algorithms using the initial controller position and final controller positions data (obtained after optimization had taken place) (see Table 2). Here is the link to the validation experiment. https://tinyurl.com/2p95ad26 and https://tinyurl.com/2p95ad26.

```
Kruskal-Wallis Test
-------------------
H statistic = 547.4398221375352
p value = 2.4937473240269875e-118
```

Fig. 21. Kruskal–Wallis Test.

```
Proposed Algorithm   : 998.4189305
Mopso Algorithm      : 1492.154173
NSGA-II Algorithm    : 1318.060786
NSGA-III Algorithm   : 1135.833079
```

Fig. 22. The final output loss median.

Interpretation:

The analysis of the final output loss data reveals important insights. Specifically, the proposed algorithm demonstrates a network final output loss mean of 936.018623, while the MOPSO algorithm exhibits the highest network final output loss mean of 1459.347972. This comparison underscores the potential differences in performance among these four controller placement algorithms based on the network's final output loss. Initially, a parametric test was conducted on the sample data. However, this research identified a deviation from the assumptions of normality and homogeneity of variance. To ensure the accuracy of the analysis and prevent any misleading conclusions, the data was subsequently subjected to a non-parametric test. This approach circumvents the requirement for homoscedasticity, thereby providing a more robust and reliable assessment of the algorithms' comparative performance in terms of network final output loss (see Fig. 24).

6.1.1. *Non-parametric test of Kruskal–Wallis*

The Kruskal–Wallis test is a nonparametric method used to compare medians among different datasets, determining if a statistically significant difference exists among multiple groups. Unlike parametric tests, it does not require assumptions of normality or equal variances, making it suitable for non-normally distributed data. This test involves ranking observations within groups and computing a test statistic based on rank sums (see Fig. 21). The null hypothesis assumes equal medians for all groups, while the alternative suggests at least one group's median differs. By comparing the test's p-value to a significance level (usually 0.05), Should the p-value be lower than the chosen significance level, indicating a statistically significant result, the null hypothesis is rejected. This infers a meaningful disparity in medians among at least two groups. If the p-value is below the threshold, indicating statistical significance, this infers meaningful median disparities among groups. Further investigation can be done using post hoc tests like Pairwise

```
Proposed Algorithm and Mopso Algorithm Comparism
================================================
RanksumsResult(statistic=-17.636206316605595, pvalue=1.2989416968608584e-69)
```

Fig. 23. Proposed algorithm and MOPSO Algorithm.

```
Proposed Algorithm and NSGA-II Algorithm Comparism
==================================================
RanksumsResult(statistic=-13.538190193544098, pvalue=9.305331701826858e-42)
```

Fig. 24. Proposed algorithm and NSGA-II Algorithm.

Wilcoxon rank-sum or Conover–Iman tests, helping identify specific groups with substantial median differences (see Fig. 25).

6.1.2. Wilcoxon rank-sum pairwise method (Post Hoc-Test)

The Wilcoxon rank-sum pairwise test is a nonparametric statistical method used to compare the medians of two independent datasets for substantial statistical differences. This test involves ranking observations from both groups and calculating the sum of ranks for each group. The test statistic is then determined as the smaller of the two rank sum totals. A p-value is computed based on the distribution of the test statistic under the null hypothesis, which assumes no difference between the groups. If the calculated probability value is lower than a predetermined significance level, typically not exceeding 0.05, the null hypothesis is rejected. This implies a statistically significant divergence in medians between the two groups. The test is particularly suitable when parametric assumptions like normality or equal variances are not met, such as with non-normally distributed or unequal variance data. It is robust and capable of handling outliers and non-normality. For multiple pairwise comparisons, it is essential to adjust the significance level, often using methods like the Bonferroni correction, to control the family-wise error rate effectively.

6.1.3. Kruskal–Wallis test

This test serves as a substitute for the ANOVA test due to the violation of assumptions regarding homoscedasticity, normality, and variance. The hypotheses for this test are as follows:

Null Hypothesis (H0): There are no significant differences among the medians of network final output losses for the four algorithms at a 5% significance level.

Alternative Hypothesis (H1): At least one of the algorithms produces a network final output loss median that is significantly different from the medians of other algorithms at a 5% significance level.

Decision: The null hypothesis is rejected at the 5% significance level, indicating significant evidence. Conclusion: At least one of the algorithms yields a network final output loss median that differs significantly from the medians of other algorithms at the 5% significance level. As a result of this significant difference, the pairwise comparison of algorithms will be conducted using the Wilcoxon rank-sum pairwise method.

6.1.4. Post HOC test: Wilcoxon rank-sum pairwise method

The median will be compared for the difference. The median for the final output loss for:

The null hypothesis is rejected, indicating a significant difference between the median values of the two compared algorithms at the 5% significance level. This suggests that the proposed algorithm exhibits a significantly lower final output loss than the MOPSO algorithm at the 5% significance level.

The null hypothesis is rejected, leading to the conclusion that a significant difference exists between the median values of the two compared algorithms at the 5% significance level. This implies that the proposed algorithm demonstrates a significantly lower final output loss compared to the MOPSO algorithm at the 5% significance level.

The null hypothesis is rejected, indicating a significant difference between the median values of the two compared algorithms at the 5% significance level. This implies that the proposed algorithm exhibits a significantly lower final output loss than the MOPSO algorithm at the 5% significance level.

In summary, the thorough and comprehensive statistical analysis has consistently yielded the rejection of the null hypothesis, revealing a significant discrepancy among the median values of all the algorithms under comparison at a 5% significance level. This provides substantial evidence that the proposed optimization algorithm consistently achieves a substantial reduction in the final output loss when contrasted with the benchmarked algorithms. The persistent and reliable rejection of the null hypothesis through multiple iterations underscores the resilience and efficacy of the proposed algorithm in achieving a noteworthy decrease in final output loss. Consequently, the proposed algorithm emerges as a compelling and favorable option for optimization, clearly outperforming the benchmarked optimization algorithm.

7. Verification and validation of the proposed stochastic computational model

In this section of the article, the author demonstrates that the major objective of the suggested model has been accomplished in terms of its quality and credibility by demonstrating that the model has carried out the tasks that it was intended to carry out. Verification includes testing, analysis of the design, and analysis of the requirements to guarantee the best quality of the final product (Sargent, 2010). One can consider this method to have a degree of objectivity. On the other hand, the nature of the process of validation is marked by a high degree of subjectivity. It necessitates the formation of personal opinions on the degree to which a problem-solving strategy that has been proposed or developed is successful in meeting a need that is existent in the real world. The validation process includes a wide range of tasks, including user testing, prototyping, and modeling the requirements. During the planning and building of the solution that was offered, the standards were adhered to in a very strict manner. The purpose of the suggested approach is to overcome the issues that were described in the literature. Some of these challenges include scalability, high computational complexity, and the absence of an established way to learn the heuristics of combinatorial optimization during the placement of controllers in SD-WAN. When looking at the figure in 26, it is evident that the proposed stochastic computational graph model for the optimization of controller placement in SD-WAN actually fulfills its intended design aim and meets the outcomes that were anticipated. This can be seen clearly by looking at this figure, which illustrates the loss/objective function progression towards the optimal solution. In a manner analogous to this, the optimum number of controllers to be deployed is demonstrated in Fig. 27. Both of these figures provide evidence that the suggested model is being verified and that it does, in fact, achieve the goals that it was designed to do.

Fig. 26 depicts the total loss function and its convergence to the optimal solution, demonstrating that this was the model's objective when it was created. The aim of this work is to devise a method for optimizing controller placement that minimizes the sum of the losses incurred by the objective functions used in conjunction with

```
Proposed Algorithm and NSGA-III Algorithm Comparism
================================================
RanksumsResult(statistic=-11.100109469074988, pvalue=1.252903356368298e-28)
```

Fig. 25. Proposed algorithm and ANSGA-III Algorithm.

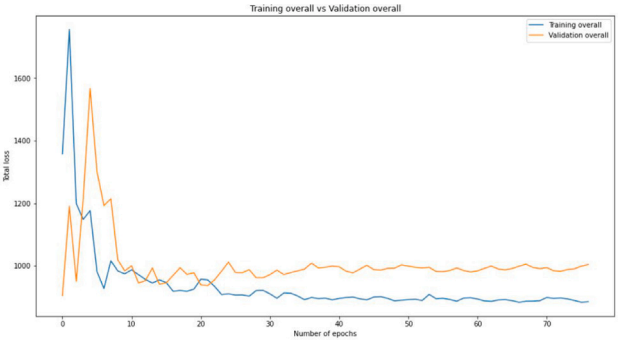


Fig. 26. Outcome of the proposed stochastic computational graph model with momentum and a learning decay rate.

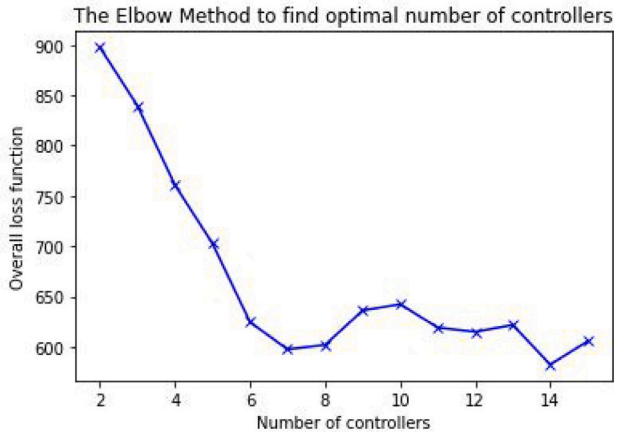


Fig. 27. Graph showing the Optimal Controller Number.

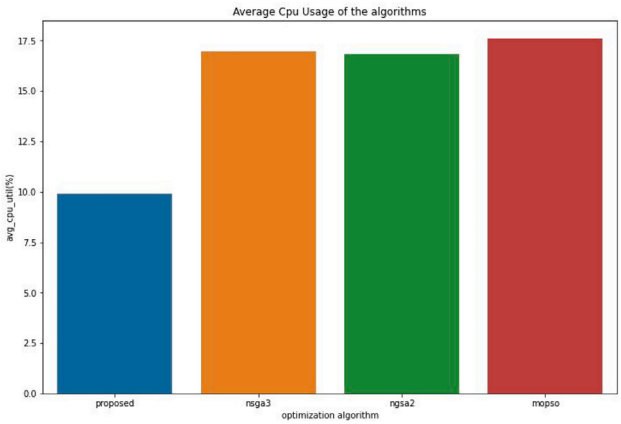


Fig. 28. Graph showing the average CPU usage of the four Algorithms.

the decision variables (controller positions). A validation or test set is used in supervised machine learning to verify the accuracy of the model on new data. As opposed to supervised learning, the suggested stochastic computational graph model uses a form of learning called reinforcement learning. Here, the model is validated by comparing its results on many test datasets. These datasets were derived from those deposited in the internet zoo topology's online repository by service providers including Bt Latin America, Bt North America, Network USA, and Tata NLD. The efficiency of the proposed stochastic computational graph model was compared to the state-of-the-art optimization algorithms (ANSGA-III, NSGA-II, and MOPSO) in the context of controller placement, which is demonstrated graphically in 28. The figure in 28, which displays total CPU consumption (the product of average CPU usage and execution time), validates the correctness of the suggested model.

8. Conclusion

The issue of controller placement in SDN has garnered significant attention in both academic and industrial circles. This challenge becomes even more critical in extensive enterprises like SD-WAN, where the optimization of numerous conflicting objectives is paramount. To tackle this complex problem, metaheuristic algorithms such as ANSGA-III, NSGA-II, and MOPSO have been proposed in the existing literature. However, these approaches, with the exception of ANSGA-III, grapple with scalability concerns—particularly when the number of objectives surpasses three. Moreover, these methods are often computationally inefficient and lack the capacity to grasp the intricate heuristics of combinatorial optimization problems like SD-WAN controller placement. In response, this study introduces a novel solution: the stochastic computational graph with an ensemble learning model. This innovative approach aims to address the challenges associated with SD-WAN controller placement.

A comprehensive comparative analysis was conducted, pitting the reviewed ANSGA-III, NSGA-II, and MOPSO against the proposed stochastic computational graph. The evaluation was based on crucial metrics, including execution time, average CPU usage, and overall CPU utilization. The findings from the experimentation unveiled the prowess of the proposed approach: the stochastic computational graph not only demonstrated enhanced computational efficiency (rapidly generating optimal controller placements) but also exhibited remarkable scalability (handling more than three conflicting objectives simultaneously). Furthermore, the solution showcased a higher level of intelligence, as exemplified by its ability to predict the optimal number of controllers required for SD-WAN deployment through the incorporation of an ensemble learning model, namely XGBoost.

In light of these compelling outcomes, the proposed stochastic computational graph with an ensemble learning model emerges as a potent remedy for the computational inefficiencies associated with existing algorithms. It proves itself adept at addressing the optimization of multiple conflicting objectives while demonstrating a keen ability to forecast the ideal number of controllers essential for SD-WAN deployment. In conclusion, the proposed approach stands out as a recommended solution for efficient and effective controller placement in SD-WAN scenarios.

CRediT authorship contribution statement

Oladipupo Adekoya: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Adel Aneiba:** Project administration, Supervision.

Data availability

No data was used for the research described in the article.

Acknowledgments

The authors extend their heartfelt appreciation to Makinde Kayode from the Engineering Department of the University of Agriculture, Abeokuta, Nigeria. His invaluable contribution in elucidating and showcasing the computational graph approach, along with its applicability in the realm of Software-Defined Networking (SDN), is sincerely acknowledged and greatly valued.

References

Adekoya, O., Aneiba, A., 2022. An adapted nondominated sorting genetic algorithm III (NSGA-III) with repair-based operator for solving controller placement problem in software-Defined Wide Area networks. IEEE Open J. Commun. Soc. 3, 888–901.

Ahmadi, V., Jalili, A., Khorramizadeh, S.M., Keshtgari, M., 2015. A hybrid NSGA-II for solving multiobjective controller placement in SDN. In: 2015 2nd International Conference on Knowledge-Based Engineering and Innovation. KBEI, IEEE, pp. 663–669.

Ahmadi, V., Khorramizadeh, M., 2018. An adaptive heuristic for multi-objective controller placement in software-defined networks. Comput. Electr. Eng. 66, 204–228.

Alouache, L., Yassa, S., Ahfir, A., 2022. A multi-objective optimization approach for SDVN controllers placement problem. In: 2022 13th International Conference on Network of the Future. NoF, IEEE, pp. 1–9.

Aravind, P., Varma, G.S., Reddy, P.P., 2022. Simulated annealing based optimal controller placement in software defined networks with capacity constraint and failure awareness. J. King Saud. Univ. Comput. Inf. Sci. 34 (8), 5721–5733.

Bagha, M.A., Majidzadeh, K., Masdari, M., Farhang, Y., 2022. Improving delay in SDNs by metaheuristic controller placement. Int. J. Ind. Electron. Control Optim. 5 (3).

Chee, J., Toulis, P., 2018. Convergence diagnostics for stochastic gradient descent with constant learning rate. In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 1476–1485.

Chen, J., Xiao, W., Li, X., Zheng, Y., Huang, X., Huang, D., Wang, M., 2022a. A routing optimization method for software-defined optical transport networks based on ensembles and reinforcement learning. Sensors 22 (21), 8139.

Chen, C., Xue, F., Lu, Z., Tang, Z., Li, C., et al., 2022b. RLMR: Reinforcement learning based multipath routing for SDN. Wirel. Commun. Mob. Comput. 2022.

De Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., Gagné, C., 2012. Deap: A python framework for evolutionary algorithms. In: Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation. pp. 85–92.

Deb, K., 2011. Multi-objective optimisation using evolutionary algorithms: an introduction. In: Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing. Springer, pp. 3–34.

Domingues, P., Marques, P., Silva, L., 2005. Resource usage of windows computer laboratories. In: 2005 International Conference on Parallel Processing Workshops. ICPPW'05, IEEE, pp. 469–476.

Evins, R., 2013. A review of computational optimisation methods applied to sustainable building design. Renew. Sustain. Energy Rev. 22, 230–245.

Gao, X., Ramezanghorbani, F., Isayev, O., Smith, J.S., Roitberg, A.E., 2020. Torchani: a free and open source pytorch-based deep learning implementation of the ANI neural network potentials. J. Chem. Inf. Model. 60 (7), 3408–3415.

Gao, C., Wang, H., Zhu, F., Zhai, L., Yi, S., 2015. A particle swarm optimization algorithm for controller placement problem in software defined network. In: International Conference on Algorithms and Architectures for Parallel Processing. Springer, pp. 44–54.

Giroire, F., Moulierac, J., Phan, T.K., 2014. Optimizing rule placement in software-defined networks for energy-aware routing. In: 2014 IEEE Global Communications Conference. IEEE, pp. 2523–2529.

Heller, B., Sherwood, R., McKeown, N., 2012. The controller placement problem. ACM SIGCOMM Comput. Commun. Rev. 42 (4), 473–478.

Hemagowri, J., Selvan, P.T., 2023. A hybrid evolutionary algorithm of optimized controller placement in SDN environment. Comput. Assist. Methods Eng. Sci..

Hock, D., Hartmann, M., Gebert, S., Jarschel, M., Zinner, T., Tran-Gia, P., 2013. Pareto-optimal resilient controller placement in SDN-based core networks. In: Proceedings of the 2013 25th International Teletraffic Congress. ITC, IEEE, pp. 1–9.

Hock, D., Hartmann, M., Gebert, S., Zinner, T., Tran-Gia, P., 2014. POCO-PLC: Enabling dynamic pareto-optimal resilient controller placement in SDN networks. In: 2014 IEEE Conference on Computer Communications Workshops. INFOCOM WKSHPS, IEEE, pp. 115–116.

Kazemian, M.M., Mirabi, M., 2022. Controller placement in software defined networks using multi-objective antlion algorithm. J. Supercomput. 1–24.

Kim, T., Zhou, X., Pendyala, R.M., 2022. Computational graph-based framework for integrating econometric models and machine learning algorithms in emerging data-driven analytical environments. Transp. A: Transp. Sci. 18 (3), 1346–1375.

Knight, S., Nguyen, H.X., Falkner, N., Bowden, R., Roughan, M., 2011. The internet topology zoo. IEEE J. Sel. Areas Commun. 29 (9), 1765–1775.

Lange, S., Gebert, S., Zinner, T., Tran-Gia, P., Hock, D., Jarschel, M., Hoffmann, M., 2015. Heuristic approaches to the controller placement problem in large scale SDN networks. IEEE Trans. Netw. Serv. Manag. 12 (1), 4–17.

Liu, Y., Gao, Y., Yin, W., 2020. An improved analysis of stochastic gradient descent with momentum. Adv. Neural Inf. Process. Syst. 33, 18261–18271.

Liu, J., Zhang, S., Kato, N., Ujikawa, H., Suzuki, K., 2015. Device-to-device communications for enhancing quality of experience in software defined multi-tier LTE-a networks. IEEE Netw. 29 (4), 46–52.

Mamun, M., Farjana, A., Al Mamun, M., Ahammed, M.S., 2022. Lung cancer prediction model using ensemble learning techniques and a systematic review analysis. In: 2022 IEEE World AI IoT Congress. AIIoT, IEEE, pp. 187–193.

Montiel, J., Mitchell, R., Frank, E., Pfahringer, B., Abdessalem, T., Bifet, A., 2020. Adaptive xgboost for evolving data streams. In: 2020 International Joint Conference on Neural Networks. IJCNN, IEEE, pp. 1–8.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A., 2017. Automatic differentiation in pytorch.

Qaffas, A.A., Kamal, S., Sayeed, F., Dutta, P., Joshi, S., Alhassan, I., 2023. Adaptive population-based multi-objective optimization in sdn controllers for cost optimization. Phys. Commun. 58, 102006.

Qi, J., Du, J., Siniscalchi, S.M., Ma, X., Lee, C.-H., 2020. On mean absolute error for deep neural network based vector-to-vector regression. IEEE Signal Process. Lett. 27, 1485–1489.

Radam, N.S., Al-Janabi, S.T.F., Jasim, K.S., 2022. Multi-controllers placement optimization in sdn by the hybrid hsa-pso algorithm. Computers 11 (7), 111.

Rashid, Z.N., Sharif, K.H., Zeebaree, S., 2018. Client/servers clustering effects on CPU execution-time, CPU usage and CPU idle depending on activities of parallel-processing-technique operations. Int. J. Sci. Technol. Res. 7 (8), 106–111.

Ros, F.J., Ruiz, P.M., 2016. On reliable controller placements in software-defined networks. Comput. Commun. 77, 41–51.

Sapkota, A., Dawadi, B., Babu, R., Joshi, C., Shashidhar, R., et al., 2022. Multi-controller placement optimization using naked mole-rat algorithm over software-defined networking environment. J. Comput. Netw. Commun. 2022.

Sargent, R.G., 2010. Verification and validation of simulation models. In: Proceedings of the 2010 Winter Simulation Conference. IEEE, pp. 166–183.

Seada, H., Deb, K., 2014. U-NSGA-III: A unified evolutionary algorithm for single, multiple, and many-objective optimization. COIN Rep. 2014022.

Stattelmann, S., Oriol, M., Gamer, T., 2014. Execution time analysis for industrial control applications. arXiv preprint arXiv:1404.0847.

Thalapala, V.S., Mohan, A., Guravaiah, K., 2022. WOACCPP: Wisdom of artificial crowds for controller placement problem with latency and reliability in SDN-WAN.

Tootoonchian, A., Ganjali, Y., 2010. Hyperflow: A distributed control plane for openflow. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, Vol. 3. pp. 10–5555.

Wang, G., Zhao, Y., Huang, J., Wang, W., 2017. The controller placement problem in software defined networking: A survey. IEEE Netw. 31 (5), 21–27.

Wu, Y., Zhou, S., Wei, Y., Leng, S., 2020. Deep reinforcement learning for controller placement in software defined network. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops. INFOCOM WKSHPS, IEEE, pp. 1254–1259.

Xiang, M., Chen, M., Wang, D., Luo, Z., 2022. Deep reinforcement learning-based load balancing strategy for multiple controllers in SDN. e-Prime-Adv. Electr. Eng. Electron. Energy 2, 100038.

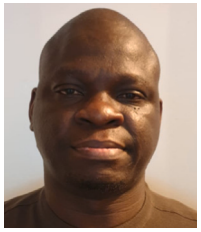
Xu, A., Sun, S., Wang, Z., Wang, X., Han, L., 2022. Multi-controller load balancing mechanism based on improved genetic algorithm. In: 2022 International Conference on Computer Communications and Networks. ICCCN, IEEE, pp. 1–8.

Yao, G., Bi, J., Li, Y., Guo, L., 2014. On the capacitated controller placement problem in software defined networks. IEEE Commun. Lett. 18 (8), 1339–1342.

Yazan, E., Talu, M.F., 2017. Comparison of the stochastic gradient descent based optimization techniques. In: 2017 International Artificial Intelligence and Data Processing Symposium. IDAP, IEEE, pp. 1–5.

Yazdinejad, A., Rabieinejad, E., Dehghantanha, A., Parizi, R.M., Srivastava, G., 2021. A machine learning-based sdn controller framework for drone management. In: 2021 IEEE Globecom Workshops. GC Wkshps, IEEE, pp. 1–6.

You, K., Long, M., Wang, J., Jordan, M.I., 2019. How does learning rate decay help modern neural networks? arXiv preprint arXiv:1908.01878.



Oladipupo Adekoya received the B.Sc. degree in Computer Science from Olabisi Onabanjo University, Nigeria, in 2007, and the M.Sc. Degree in Data Network and Security from Birmingham City University, UK, in 2018, where he is currently pursuing his Ph.D. degree with the School of Computing Engineering and Built Environment. His research interests include SDN/NFV and Artificial Intelligence. He is a Member of IET.

Adel Aneiba received the B.Sc. degree in Computer Science from the University of Benghazi in Libya, the M.Sc. degree in e-commerce from Staffordshire University in the year 2003, and the Ph.D. degree in Computing in the year 2008. He is an Associate Professor in the Internet of Things (IoT) at Birmingham City University, UK. His research interests include IoT, computer network simulation, evaluation, optimization and block-chain. He is a Member of IET.