network

MDPI

Article

Dynamic Multi-Objective Controller Placement in SD-WAN: A GMM-MARL Hybrid Framework

Abdulrahman M. Abdulghani [1,*], Azizol Abdullah [1], A. R. Rahiman [1], Nor Asilah Wati Abdul Hamid [1,2] and Bilal Omar Akram [3,4]

[1] Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Selangor, Malaysia
[2] Institute for Mathematical Research, Universiti Putra Malaysia (UPM), Serdang 43400, Selangor, Malaysia
[3] Department of Computer and Communication Systems Engineering, Faculty of Engineering, Universiti Putra Malaysia (UPM), Serdang 43400, Selangor, Malaysia
[4] Wireless and Photonics Networks Research Centre of Excellence (WiPNET), Faculty of Engineering, Universiti Putra Malaysia (UPM), Serdang 43400, Selangor, Malaysia
* Correspondence: abdulrahman.m.abdulghani@gmail.com

Abstract

Modern Software-Defined Wide Area Networks (SD-WANs) require adaptive controller placement addressing multi-objective optimization where latency minimization, load balancing, and fault tolerance must be simultaneously optimized. Traditional static approaches fail under dynamic network conditions with evolving traffic patterns and topology changes. This paper presents a novel hybrid framework integrating Gaussian Mixture Model (GMM) clustering with Multi-Agent Reinforcement Learning (MARL) for dynamic controller placement. The approach leverages probabilistic clustering for intelligent MARL initialization, reducing exploration requirements. Centralized Training with Decentralized Execution (CTDE) enables distributed optimization through cooperative agents. Experimental evaluation using real-world topologies demonstrates a noticeable reduction in the latency, improvement in network balance, and significant computational efficiency versus existing methods. Dynamic adaptation experiments confirm superior scalability during network changes. The hybrid architecture achieves linear scalability through problem decomposition while maintaining real-time responsiveness, establishing practical viability.

Keywords: software-defined WAN; controller placement; multi-agent reinforcement learning; gaussian mixture model; dynamic optimization; multi-objective optimization; CRITIC method

check for updates

Academic Editors: Dapeng Dong and Jun (John) Huang

Received: 6 September 2025
Revised: 2 November 2025
Accepted: 7 November 2025
Published: 11 November 2025

Citation: Abdulghani, A.M.; Abdullah, A.; Rahiman, A.R.; Abdul Hamid, N.A.W.; Akram, B.O. Dynamic Multi-Objective Controller Placement in SD-WAN: A GMM-MARL Hybrid Framework. *Network* **2025**, *5*, 52. https://doi.org/10.3390/network5040052

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The paradigmatic shift towards Software-Defined Wide Area Networks (SD-WANs) has fundamentally transformed enterprise network architectures, enabling unprecedented flexibility, cost efficiency, and centralized management capabilities. By decoupling the control plane from the data plane, SD-WANs empower organizations to dynamically optimize network performance, implement sophisticated traffic policies, and rapidly adapt to evolving business requirements [1]. However, this architectural transformation introduces a critical optimization challenge: the strategic placement of controllers within distributed network infrastructures [2]. The Controller Placement Problem (CPP) represents one of the most fundamental challenges in SD-WAN deployment, directly impacting network performance, scalability, and operational efficiency [3]. As enterprise networks continue

to expand in both scale and complexity, the limitations of single-controller architectures have become increasingly apparent [4]. Single points of failure, scalability bottlenecks, and performance degradation under high-traffic loads necessitate distributed multi-controller architectures that can maintain service quality while ensuring fault tolerance and load distribution [5]. The criticality of optimal controller placement is underscored by its direct correlation with key performance indicators including propagation latency, load balancing effectiveness, fault tolerance, and overall network responsiveness. Suboptimal controller placement decisions can result in significant performance degradation, increased operational costs, and reduced user experience quality. Conversely, well-optimized controller placement strategies can deliver substantial improvements in network efficiency, resource utilization, and service reliability [6].

Ongoing SD-WAN deployments operate in highly dynamic environments characterized by frequent topology changes, varying traffic patterns, evolving service requirements, and unpredictable network events [1,3]. Traditional static controller placement approaches, which assume fixed network conditions and predetermined traffic patterns, are fundamentally inadequate for addressing the adaptive requirements of modern enterprise networks [7]. These static methodologies typically optimize controller placement based on historical network data or simplified network models, failing to account for the inherent dynamism of real-world network operations. The limitations of static approaches manifest in several critical areas. First, static placement strategies cannot adapt to changing network conditions, resulting in suboptimal performance as network characteristics evolve over time. Second, these approaches typically focus on single optimization objectives, such as latency minimization or cost reduction, without considering the complex trade-offs between multiple competing performance metrics. Third, static methods often exhibit poor scalability characteristics, becoming computationally intractable for large-scale network deployments [8]. Dynamic controller placement presents additional challenges beyond those addressed by static methods. The multi-objective nature of the optimization problem requires balancing conflicting objectives such as minimizing average control latency while maintaining load balance and ensuring fault tolerance [9]. The temporal dimension adds complexity, as optimal placement decisions must consider both current network conditions and anticipated future states [4–8]. Furthermore, the distributed nature of multi-controller architectures introduces coordination challenges, requiring sophisticated mechanisms to ensure consistent and coherent network-wide optimization. The scalability challenge is particularly acute in contemporary enterprise environments, where networks may encompass hundreds or thousands of nodes distributed across multiple geographic locations. Traditional optimization approaches often exhibit exponential computational complexity, making them impractical for large-scale deployments where real-time adaptation is essential. Therefore, the need for solutions that can achieve near-optimal performance while maintaining computational tractability represents a significant research challenge [8–10].

Existing controller placement methodologies face limitations due to computational complexity, which render them impractical for large-scale or real-time applications, while heuristic algorithms offer improved efficiency but sacrifice optimality. Reinforcement learning (RL) approaches show superior adaptability but have limitations such as cold-start problems, poor sample efficiency, and focusing on single-agent scenarios.

To address the limitations of existing approaches, this paper proposes a novel hybrid framework that integrates Gaussian Mixture Model (GMM) clustering with Multi-Agent Reinforcement Learning (MARL) for dynamic controller placement optimization. The GMM-MARL framework leverages the complementary strengths of probabilistic clustering and adaptive learning to achieve superior performance across multiple optimization objectives while maintaining computational efficiency suitable for large-scale deployments. This

work makes several significant contributions to the field of dynamic controller placement in SD-WAN environments:

- Novel Hybrid Architecture: The integration of GMM clustering with MARL represents the first systematic approach to combining probabilistic clustering with multi-agent reinforcement learning for controller placement optimization.
- Adaptive Multi-Objective Optimization: The incorporation of the CRITIC method for dynamic objective weighting provides a principled approach to multi-objective optimization that automatically adapts to network characteristics without requiring manual parameter tuning.
- Scalable Distributed Learning: The CTDE-based MARL implementation enables distributed optimization that scales linearly with network size while maintaining cooperative behaviour between agents.
- Comprehensive Evaluation Framework: The development of a multi-metric evaluation framework that considers latency, load balancing, inter-controller communication, and dynamic adaptation capabilities provides a more holistic assessment methodology for controller placement algorithms.
- Real-World Validation: Extensive experimental evaluation using real-world network topologies from the Internet Topology Zoo demonstrates the practical applicability of the proposed approach. The evaluation encompasses both static performance comparison and dynamic adaptation scenarios, providing comprehensive validation of the framework's effectiveness.

The remainder of this article proceeds as follows. Section 2 surveys controller-placement literature; Section 3 details the proposed GMM-MARL framework; Section 4 reports the results; Section 5 discusses limitations and future work; and Section 6 concludes with implications for SD-WAN deployment.

2. Background and Related Works

2.1. Software-Defined Networking and Controller Placement Fundamentals

Software-Defined Networking (SDN) has revolutionized network management by decoupling the control plane from the data plane, enabling centralized network control and programmability [1], see Figure 1.

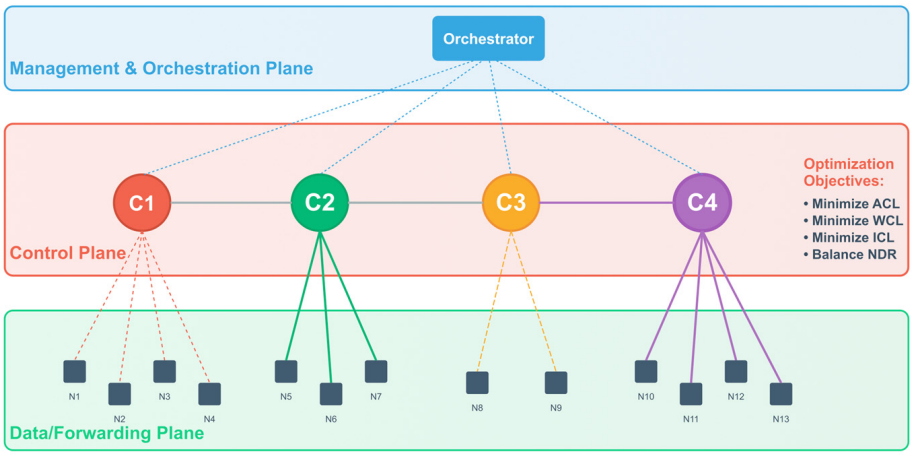


Figure 1. SDN/SD-WAN Planes Architecture.

The evolution towards Software-Defined Wide Area Networks (SD-WANs) has extended these principles to enterprise networks, providing enhanced flexibility, cost reduction, and simplified management [2]. However, the distributed nature of modern

networks necessitates multiple controllers to ensure scalability, fault tolerance, and acceptable performance, leading to the critical CPP [3]. The CPP fundamentally addresses three interconnected questions: determining the optimal number of controllers, identifying their strategic placement locations, and establishing efficient switch-to-controller mappings [4]. Traditional approaches have predominantly focused on static placement strategies, assuming fixed network topologies and predictable traffic patterns. However, the dynamic nature of contemporary SD-WAN deployments, characterized by frequent topology changes, varying traffic loads, and evolving service requirements, has exposed the limitations of static methodologies [5].

2.2. Static Controller Placement Approaches

Early research in controller placement primarily employed static optimization techniques, treating the problem as a facility location or graph partitioning challenge. Mathematical programming approaches, including Integer Linear Programming (ILP) and mixed-integer programming (MIP), have been extensively studied for optimal controller placement [6]. These deterministic methods guarantee globally optimal solutions for small-scale networks but suffer from computational complexity limitations that render them impractical for large-scale deployments. Heuristic algorithms have emerged as practical alternatives to exact optimization methods. Clustering-based approaches, such as K-means and hierarchical clustering, have been widely adopted due to their computational efficiency and intuitive network partitioning capabilities [7]. The probabilistic GMM clustering algorithm has served as a foundation for numerous controller placement studies, offering polynomial-time complexity while achieving near-optimal solutions for certain network topologies [7,8]. Recent advances in clustering-based methods include the Greedy Optimized K-Means Algorithm (GOKA), which combines greedy optimization with traditional clustering to minimize propagation latency while maintaining computational efficiency [9]. GOKA iteratively merges clusters based on latency improvements, providing superior performance compared to standard K-means approaches. However, these static methods remain fundamentally limited by their inability to adapt to dynamic network conditions.

2.3. Dynamic Controller Placement and Adaptation

The recognition that network conditions in SD-WAN environments are inherently dynamic has motivated research into adaptive controller placement strategies. Dynamic approaches aim to continuously optimize controller placement and switch assignments in response to changing network conditions, including traffic fluctuations, topology modifications, and performance degradation [10]. Traffic-aware controller placement represents one of the earliest dynamic approaches, utilizing historical traffic patterns and predictive models to anticipate network changes [11]. These methods employ time-series analysis and machine learning techniques to forecast traffic demands and proactively adjust controller placements. However, the accuracy of such approaches is fundamentally limited by the predictability of network traffic patterns. Event-driven adaptation mechanisms have been proposed to address the limitations of predictive approaches. These systems monitor network events, such as link failures, congestion, and topology changes, triggering controller reassignment when predefined thresholds are exceeded [12]. While more responsive than predictive methods, event-driven approaches often result in reactive rather than proactive optimization, potentially leading to temporary performance degradation during transition periods. Load balancing in dynamic environments presents additional complexity, as controller utilization can vary significantly over time. Adaptive load balancing algorithms continuously monitor controller workloads and redistribute switch assignments to maintain balanced resource utilization [13]. These approaches often employ feedback control

mechanisms to achieve stable load distribution while minimizing disruption to ongoing network operations.

2.4. Machine Learning and Artificial Intelligence Approaches

The application of machine learning and artificial intelligence techniques to controller placement has gained significant momentum in recent years, driven by their ability to handle complex, high-dimensional optimization problems and adapt to dynamic environments [14]. Supervised learning approaches have been employed to learn optimal placement patterns from historical network data, enabling prediction of optimal controller configurations for new network scenarios [15]. Reinforcement Learning (RL) has emerged as a particularly promising approach for dynamic controller placement due to its ability to learn optimal policies through interaction with the environment. Deep Q-Networks (DQN) have been applied to controller placement problems, demonstrating the ability to discover near-optimal placements without requiring explicit optimization objectives [16]. The DQN framework enables agents to learn from experience, gradually improving placement decisions based on observed network performance. Advanced RL techniques have further enhanced the applicability of learning-based approaches to controller placement. The Multi-Objective Optimization-Oriented Rainbow Deep Q-Network (MOOO-RDQN) integrates multiple DQN enhancements, including double Q-learning, prioritized experience replay, dueling networks, multi-step learning, and noisy networks [17]. MOOO-RDQN demonstrates significant improvements in both convergence speed and solution quality compared to standard DQN approaches, achieving up to a 42% reduction in average latency and a 59% improvement in worst-case latency scenarios. Multi-Agent Reinforcement Learning (MARL) represents a natural extension of single-agent approaches, enabling distributed optimization through cooperative or competitive agent interactions [18]. MARL approaches model individual controllers as autonomous agents that learn to coordinate their actions to optimize global network performance. The Centralized Training with Decentralized Execution (CTDE) paradigm has proven particularly effective, allowing agents to learn cooperative behaviours during training while maintaining autonomous operation during deployment [19].

2.5. Multi-Objective Optimization in Controller Placement

Real-world controller placement scenarios invariably involve multiple, often conflicting objectives that must be balanced to achieve acceptable overall performance. Traditional approaches typically focus on single objectives, such as minimizing average latency or maximizing load balancing, potentially leading to suboptimal solutions when multiple criteria are considered simultaneously [20]. Multi-objective optimization techniques have been developed to address this limitation by explicitly considering trade-offs between competing objectives. Pareto optimization approaches seek to identify the set of non-dominated solutions that represent optimal trade-offs between different performance metrics [21]. These methods enable network operators to select controller placements that best align with their specific priorities and constraints. Weighted sum approaches provide a simpler alternative to Pareto optimization by combining multiple objectives into a single composite metric. However, the selection of appropriate weights often requires domain expertise and may not adequately capture the relative importance of different objectives under varying network conditions [22]. Adaptive weighting mechanisms, such as the criteria importance through intercriteria correlation (CRITIC) method, have been proposed to automatically determine objective weights based on metric variability and interdependence [23]. MuZero-based intelligent agent approaches model proposed controller placement as a strategic interaction between multiple decision-makers, each seeking to optimize their local objectives while

considering the actions of others [24]. These methods can capture competitive scenarios where different network domains or service providers seek to optimize their individual performance metrics while sharing common infrastructure resources.

2.6. Hybrid Approaches and Advanced Techniques

The limitations of individual methodologies have motivated the development of hybrid approaches that combine the strengths of multiple techniques. Clustering-based initialization followed by optimization refinement represents a common hybrid strategy, leveraging the computational efficiency of clustering methods while achieving the solution quality of optimization techniques [25]. Machine learning enhanced clustering approaches integrate learning mechanisms into traditional clustering algorithms to improve their adaptability to network characteristics. These methods employ historical performance data to tune clustering parameters and adapt partitioning strategies to specific network topologies [26]. Probabilistic clustering methods, such as GMM, offer enhanced flexibility compared to deterministic clustering approaches by modelling cluster membership as probabilistic assignments rather than discrete decisions [8,27–29]. GMM-based approaches can naturally handle overlapping clusters and uncertain node assignments, providing more robust solutions in dynamic environments where network characteristics may vary over time. The integration of probabilistic clustering with reinforcement learning represents a recent advancement in hybrid methodologies. These approaches leverage GMM clustering to provide intelligent initialization for RL agents, reducing exploration requirements and accelerating convergence [30]. The combination enables the benefits of both probabilistic modelling and adaptive learning, resulting in superior performance compared to individual approaches.

3. Methodology

This section presents our proposed GMM-MARL framework, which integrates Gaussian Mixture Model clustering with Multi-Agent Reinforcement Learning to achieve dynamic, multi-objective controller placement optimization. The methodology encompasses GMM-MARL framework implementation with CRITIC-based metric weighting and cooperative learning mechanisms. Our approach addresses the limitations of existing static methods by providing real-time adaptability to network changes while maintaining computational efficiency suitable for large-scale deployments. Figure 2 represents the proposed GMM–MARL hybrid model.

3.1. Problem Formulation

Consider an SD-WAN represented as an undirected graph $G = (V,E)$, where $V = \{v1, v2, \ldots, vn\}$ denotes the set of network nodes (switches/devices) and E represents the communication links between them. The controller placement problem seeks to determine optimal positions $C = \{c1, c2, \ldots, ck\}$ for k controllers from candidate locations to minimize multiple competing objectives simultaneously. The multi-objective optimization objective to overcome the problem is formulated as:

$$\min_{C \subseteq V} \{f1(C), f2(C), f3(C), f4(C)\}$$

where

$f1(C)$: Average Control Latency (ACL)—delay between nodes and controllers;
$f2(C)$: Worst-case Control Latency (WCL)—maximum latency within any cluster;
$f3(C)$: Inter-Controller Latency (ICL)—communication delay between controllers;
$f4(C)$: Node Distribution Ratio (NDR)—load balancing metric across controllers,
Subject to constraints:

- Each node $vi \in V$ must be assigned to exactly one controller;
- Controller capacity: $\sum vi \in Sj\ di \leq Capj$, where Sj is the set of nodes assigned to controller cj, di is the demand of node vi, and $Capj$ is the capacity of controller cj;
- Latency bound: $l(vi,cj) \leq Lmax$ for all node-controller pairs.

The proposed approach addresses this through a two-phase optimization framework combining probabilistic clustering with reinforcement learning-based adaptation, the following Table 1a,b represent the denotations and key hyperparameters used in our implementation of the method.

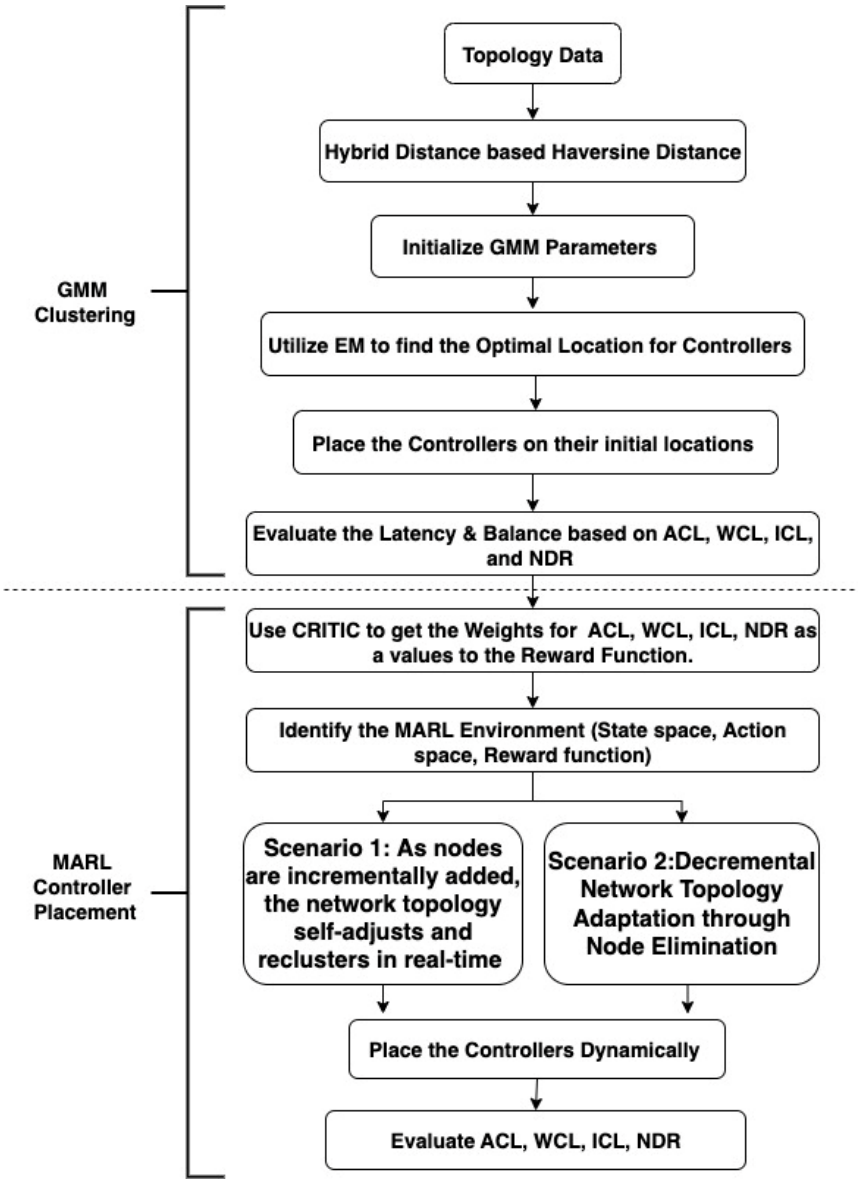


Figure 2. Represents the proposed GMM–MARL hybrid model for dynamic controller placement.

This formulation provides the mathematical foundation for our GMM-MARL approach, where the dynamic nature of the problem is captured through time-dependent variables and the multi-objective optimization is balanced through CRITIC-weighted objective functions.

Table 1. (**a**) Mathematical Notation and Parameters. (**b**) hyperparameters used in the implementation.

(**a**)

Symbol	Description	Domain
G(t)	Time-varying network graph at time *t*	Graph
N(t)	Set of network nodes at time *t*	Set
K	Total number of clusters/controllers	$\mathbb{N}+$
C(t)	Set of controller locations at time *t*	Set
E(t)	Set of network links at time *t*	Set
dNA(i,j)	Network-aware hybrid distance between nodes *i,j*	$\mathbb{R}+$
ACL	Average Cluster Latency	$\mathbb{R}+$ [ms]
WCL	Worst-case Cluster Latency	$\mathbb{R}+$ [ms]
ICL	Inter-Controller Latency	$\mathbb{R}+$ [ms]
NDR	Node Distribution Ratio	$\mathbb{R}+$
Wm	CRITIC-based weight for metric *m*	[0,1]
α, β, γ, δ	Hybrid distance metric weights	[0,1], $\Sigma = 1$
μk, Σk	GMM cluster parameters (mean, covariance)	$\mathbb{R}$d, $\mathbb{R}$d $\times$ d
πk	GMM mixing coefficients	[0,1]
k	Cluster/controller index	{1, . . .,K}
γik	MARL responsibility values	[0,1]
Qt(s,a)	Q-value function at time *t*	$\mathbb{R}$
rt	Reward at time step *t*	$\mathbb{R}$
st	Network state at time *t*	State space
at	Agent action at time *t*	Action space
ε	Convergence threshold	$\mathbb{R}+$

(**b**)

Category	Parameter	Symbol	Value	Description
GMM	Max Iterations	–	100	EM algorithm iterations
GMM	Threshold	ε	0.001	EM convergence criterion
Q-Network	Learning Rate	α	0.001	Adam optimizer learning rate
Q-Network	Discount Factor	γ	0.95	Future reward discount
Q-Network	Batch Size	–	32	Mini-batch for training
Q-Network	Replay Buffer	B_i	10,000	Experience buffer capacity
Exploration	Initial ε	ε_max	1.0	Starting exploration rate
Exploration	Final ε	ε_min	0.01	Minimum exploration rate
Exploration	Decay Rate	λ_ε	0.995	Exponential decay factor
Network	Hidden Layer 1	H1	256	First layer neurons
Network	Hidden Layer 2	H2	128	Second layer neurons
Network	Target Update	–	100	Target network sync frequency
Training	Max Episodes	–	1000	Training episode limit
Training	Convergence	ε_conv	0.001	Convergence threshold

3.2. Network-Aware Hybrid Distance Metric

Traditional controller placement methodologies rely predominantly on simplistic distance metrics. The proposed method leverages the hybrid distance metric that utilized in [8] integrates four critical network dimensions to provide a comprehensive assessment of node relationships in SD-WAN environments. The hybrid distance metric $dNA(i,j)$ between any two nodes i and j is formulated as (1):

$$dNA(i,j) = \alpha \cdot dgeo(i,j) + \beta \cdot dlat(i,j) + \gamma \cdot dtopo(i,j) + \delta \cdot R(i,j) \quad (1)$$

where

- α, β, γ, δ are adaptive weight parameters satisfying $\alpha + \beta + \gamma + \delta = 1$;
- dgeo(i,j) represents the geodesic distance between nodes i and j calculated using the Haversine formula;
- dlat(i,j) denotes the propagation latency determined by physical distance divided by signal propagation speed;
- dtopo(i,j) is the topological distance measured as the minimum hop count between nodes in the network graph;
- R(i,j) represents the reliability factor ($0 \leq R(i,j) \leq 1$) measuring link quality based on historical performance metrics.

3.3. Gaussian Mixture Model Framework

The GMM-based clustering strategy extends classical unsupervised learning to accommodate spatial distributions and network-specific quality measures. This probabilistic framework enables multi-objective optimization of control plane design with respect to latency, scalability, load balancing, and fault tolerance. A GMM with K components define the probability density function of a network node $x \in \mathbb{R}d$ as (2):

$$p(x) = \Sigma(k = 1 \text{ to } K)\ \pi k \cdot N(x|\mu k, \Sigma k) \quad (2)$$

where

- $\pi k \in [0,1]$ is the mixing coefficient for component k, satisfying $\Sigma \pi k = 1$
- $\mu k \in \mathbb{R}d$ is the mean vector (centre) of the $k-th$ Gaussian distribution;
- $\Sigma k \in \mathbb{R}d \times d$ is the covariance matrix capturing distribution spread;
- $N(x|\mu k, \Sigma k)$ denotes the multivariate normal distribution.

Expectation-Maximization Algorithm: Parameter estimation employs the iterative EM algorithm:

1. E-step: Compute responsibilities γik representing the probability of node i belonging to controller cluster k, as (3):

$$\gamma ik = (\pi k \cdot N(xi|\mu k, \Sigma k)) \ / \ \Sigma(j = 1 \text{ to } K)\ \pi j \cdot N(xi|\mu j, \Sigma j) \quad (3)$$

2. M-step: Update model parameters based on computed responsibilities, as represented in (4)–(6):

$$\mu k^{t+1} = \left(\frac{1}{Nk}\right) \Sigma(i = 1 \text{ to } N) \gamma ik \cdot xi \quad (4)$$

$$\Sigma k^{t+1} = \left(\frac{1}{Nk}\right) \Sigma(i = 1 \text{ to } N) \gamma ik \cdot (xi - \mu k)(xi - \mu k)^T \quad (5)$$

$$\pi k^{t+1} = \frac{Nk}{N} \quad (6)$$

where $Nk = \Sigma(i = 1 \text{ to } N)\ \gamma ik$ represents the effective sample size for cluster k.

Convergence Monitoring: The algorithm monitors convergence through the log-likelihood function, as seen in (7):

$$log\ L(\theta)\ =\ \Sigma(i=1\ to\ N)\ log(\Sigma(k=1\ to\ K)\ \pi k{\cdot}N(xi|\mu k,\Sigma k)) \tag{7}$$

terminating when $|log\ L(t)\ -\ log\ L(t-1)| <\ \varepsilon$.

This hybrid approach enables more accurate representation of real-world network relationships and facilitates optimal controller placement decisions. The iterative nature of the EM algorithm ensures convergence to locally optimal solutions while maintaining computational efficiency suitable for large-scale SD-WAN deployments statically, see Algorithm 1.

Algorithm 1 Network-Aware GMM Controller Placement

Input: Node coordinates {φi, λi} for i = 1, . . ., N
Input: Weight parameters α, β, γ, δ
Input: Earth radius r, propagation speed v, base cost c0, cost factor c1, decay factor κ
Input: Number of clusters K, distance matrix DNA ∈ ℝN × N
Output: Controller positions μk and node-cluster assignments γik
Initialize Environment:
    DNA(i,j) ← α·d′geo(i,j) + β·d′lat(i,j) + γ·d′topo(i,j) − δ·R′(i,j)
  end for
  return DNA
Initialize GMM parameters: μk, Σk ← 1, πk ← 1/K
repeat
  E-step:
  for each node i and cluster k do
    Compute responsibility:
      γik ← (πk·N(xi | μk,Σk))/(Σj = 1^K πj·N(xi | μj,Σj))
  end for
  M-step:
  for each cluster k do
    Nk ← Σi = 1^N γik
    Update mean:
      μk ← (1/Nk)·Σi = 1^N γik·xi
    Update covariance:
      Σk ← (1/Nk)·Σi = 1^N γik·(xi − μk)(xi − μk)^T
    Update mixing coefficient:
      πk ← Nk/N
  end for
  Evaluate log-likelihood L(θ) and check convergence
until Change in L(θ) < ε
Assign each node i to cluster k = arg maxk γik
return Controller positions μk and assignments γik

3.4. Performance Metrics

Following GMM clustering and initial controller placement, we compute four key performance metrics:

- *Average Control Latency* (*ACL*)*:* quantifies the mean communication delay within controller domains, serving as a primary indicator of network responsiveness, as in (8):

$$ACL = \frac{1}{n}\sum_{i=1}^{n} l = (vi, C\ assign(vi)) \tag{8}$$

Here, $C\ assign(vi)$ denotes the controller to which node vi is assigned, where the assignment maps each node to its nearest controller based on the network-aware distance metric defined in Equation (1).

- Worst-case Control Latency (WCL): captures the maximum delay scenarios, ensuring that performance optimization addresses edge cases that could impact critical applications, as seen in (9):

$$WCL = \min_{j \in \{1,\ldots,k\}} \max_{vi \in Sj} l(vi, cj) \tag{9}$$

- Inter-Controller Latency (ICL): measures coordination overhead between controllers, directly affecting the system's ability to maintain consistent network state and implement coordinated policies, as in (10):

$$ICL = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} l(ci, cj) \tag{10}$$

- Node Distribution Ratio (NDR): evaluates load balancing effectiveness, preventing controller overload and ensuring scalable resource utilization, see (11):

$$NDR = \frac{maxj \mid Sj \mid}{minj \mid Sj \mid} \tag{11}$$

where $\mid Sj \mid$ denotes the number of nodes assigned to controller cj.

The transformation of these metrics into actionable optimization drivers represents a significant methodological advancement. Rather than treating these metrics as static evaluation criteria, our framework employs the CRITIC method to dynamically assess their relative importance based on current network conditions.

3.5. CRITIC-Based Weight Assignment

To balance multiple objectives effectively, we employ the Criteria Importance Through Intercriteria Correlation (CRITIC) method [31] to determine objective weights based on metric variability and inter-correlations.

- Step1: For each metric $Xm \in \{ACL, WCL, ICL, NDR\}$, compute the normalized value, as (12):

$$X^{-}m = \frac{Xm - \min(Xm)}{\max(Xm) - \min(Xm)} \tag{12}$$

- Step 2: Calculate standard deviation, represented in (13):

$$\sigma Xm = \sqrt{\frac{1}{s} \sum_{t=1}^{s} (Xmt - \mu m)2} \tag{13}$$

where s is the number of samples and μm is the mean of metric Xm.

- Step 3: Construct correlation matrix rmn, as in (14)

$$rmn = \frac{Cov(Xm, Xn)}{\sigma Xm \cdot \sigma Xn} \tag{14}$$

- Step 4: Compute unnormalized weights, see (15)

$$Wm = \sigma Xm \cdot \sum_{n=1, n \neq m}^{4} (1 - \mid rmn \mid) \tag{15}$$

- Step 5: Normalize weights, as seen in (16)

$$W^{-}m = \frac{Wm}{\sum_{k=1}^{4} Wk} \tag{16}$$

These weights capture both the information content (variability) and uniqueness (low correlation) of each metric, ensuring balanced optimization in the subsequent MARL phase.

3.6. MARL-Based Dynamic Optimization

The dynamic controller placement problem is formulated as a decentralized multi-agent system where each SDN controller operates as an autonomous learning agent, enabling scalable and adaptive optimization without requiring centralized coordination during deployment. In this architecture, each agent $i \in \{1, 2, \ldots, k\}$ represents a controller responsible for managing its assigned network nodes, monitoring local network conditions, making placement and assignment decisions, and coordinating with neighbouring controllers. The system operates under the CTDE paradigm use a centralized critic during training and decentralized actors during execution, following the framework introduced in [32], which enables agents to learn cooperative behaviours during the training phase while maintaining autonomous decision-making capabilities during deployment, thus balancing global optimization with computational efficiency, as shown in Figure 3.

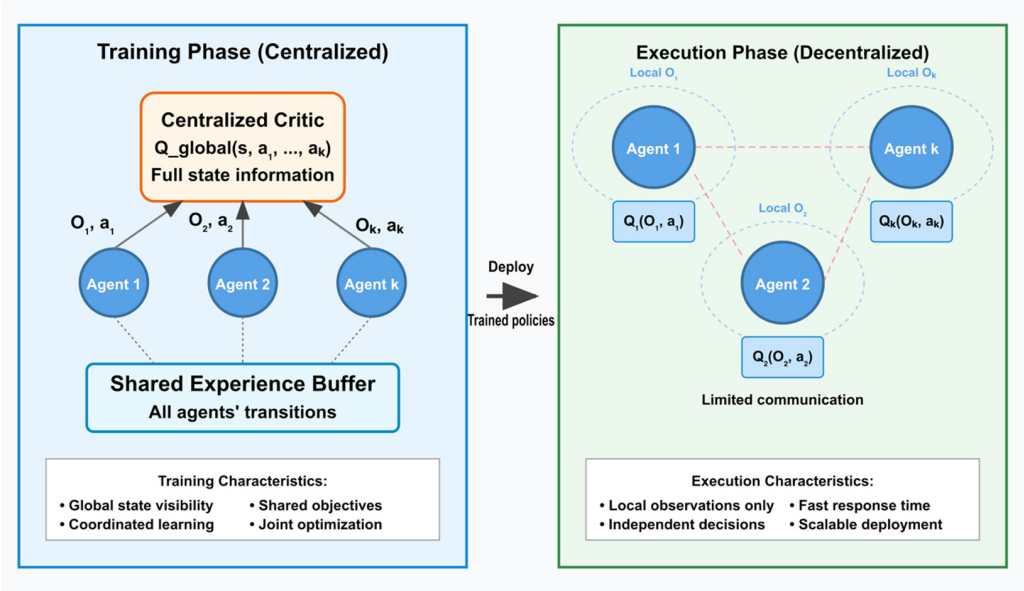


Figure 3. Illustrates the centralized training with decentralized execution (CTDE) framework.

The global network state at time t encompasses the complete information necessary for optimal decision-making, including the set of active nodes with their positions and traffic demands, current controller positions in the network, propagation delays between all node-controller pairs, controller utilization rates, and inter-node communication patterns. This comprehensive state representation is formally expressed as (17):

$$St = (Nt, Ct, Lt, Ut, Tt) \tag{17}$$

where $Nt = \{n_1{}^t, n_2{}^t, \ldots, n_m{}^t\}$ represents active nodes, $Ct = \{c_1{}^t, c_2{}^t, \ldots, c_k{}^t\}$ denotes controller positions, $Lt \in \mathbb{R}^{mxk}$ is the latency matrix with $L_\{ij\}t$ representing propagation delay from node i to controller j, $Ut = \{u_1{}^t, u_2{}^t, \ldots, u_k{}^t\}$ contains controller utilization rates where $u_ft = current_load_j/capacity_j$, *and* $T_t \in \mathbb{R}^{mxm}$ captures the traffic matrix representing inter-node communication patterns. Active nodes are defined as network elements (SDN Switches/Routers) that are currently operational and generating traffic, excluding any failed, inactive, or maintenance-mode nodes. Each active node ni^t, is characterized by:

- Position coordinates (xi, yi) in the network topology;

- Traffic demand di^t, measured in Mbps;
- Connectivity status indicating reachability to controllers;
- Processing capacity for flow rule installation.

3.6.1. Observation and Action Spaces

Due to the inherent scalability challenges and distributed nature of SD-WANs, each agent maintains partial observability of the global state through a structured observation model that captures local, neighbourhood, and essential global information. This partial observability model ensures computational tractability while providing sufficient information for effective decision-making. The observation space for each agent i at time t is hierarchically structured to include local observations comprising nodes directly managed by the agent along with their latency measurements and utilization metrics, neighbourhood observations capturing the state of nearby controllers within a predefined communication radius including their positions and inter-controller latencies, and global indicators providing system-wide performance trends and topology changes that affect overall network behaviour, see Figure 4 and Formula (18).

$$O_i^t = \left\{ O_{local}^i, O_{neighbor}^i, O_{global}^i \right\} \tag{18}$$

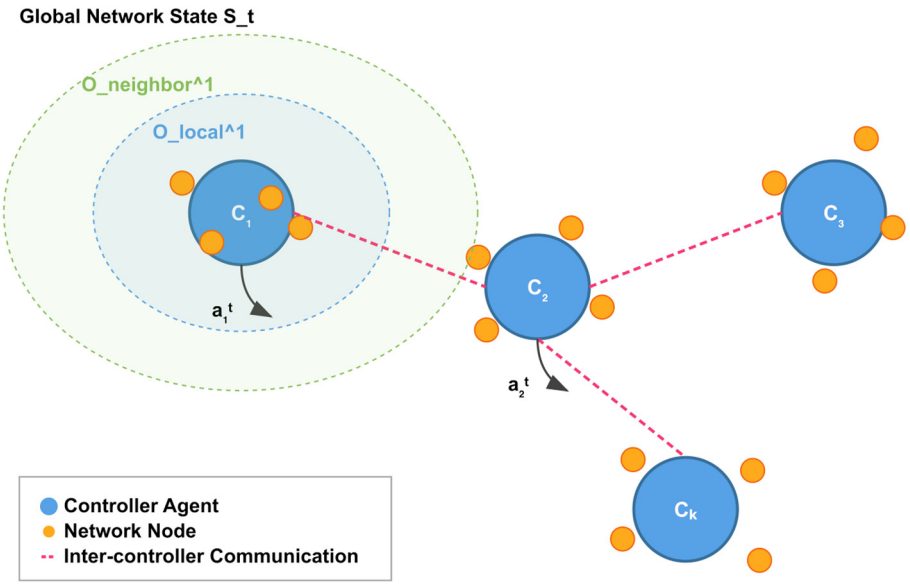


Figure 4. Depicts multi-agent system architecture for dynamic controller placement.

The action space for each agent consists of three primary components that enable comprehensive control over the network configuration. Position adjustment actions allow controllers to relocate to alternative positions from a candidate set based on traffic density and geographical distribution patterns, constrained by physical infrastructure availability. Assignment modification actions enable the redistribution of nodes between controllers to maintain load balance, represented as binary decision vectors indicating whether each node should be maintained or transferred to neighbouring controllers. Coordination actions facilitate information sharing and synchronization between agents, ensuring consistent network-wide optimization. The complete action space is formulated as (19):

$$a_i^t = \left(a_{position}^i, a_{assign}^i, a_{coordinate}^i \right) \tag{19}$$

where $a^i_{position}$ represents controller repositioning decisions selected from the candidate location set $V_{\{\text{candidate}\}}$, $a^i_{assign} = \{0,1\}^{\{|N_it|\}}$ denotes binary node assignment decisions, and $a^i_{coordinate}$ encompasses communication and synchronization actions with neighbouring agents.

3.6.2. Reward Engineering with CRITIC Weights

The reward function design is critical for guiding agent behaviour toward optimal network performance while maintaining stability and avoiding local optima. The reward structure combines immediate performance feedback with long-term optimization objectives, leveraging the CRITIC-derived weights to ensure balanced consideration of all performance metrics. The immediate reward component captures the instantaneous change in network performance, providing rapid feedback for agent learning and enabling quick responses to network dynamics; see (20):

$$R^t_{immediate} = -\sum_{i=1}^{4} W^- i \cdot \Delta X^t_i \tag{20}$$

where $\Delta X^t_i = X^t_i - X^{\{t-1\}}_i$ represents the temporal change in each performance metric, and negative values indicate improvement since lower metric values correspond to better performance. The global reward signal evaluates the overall network state relative to theoretical bounds, normalizing each metric to a comparable scale and aggregating them according to their CRITIC-determined importance. This normalization ensures that metrics with different units and ranges contribute proportionally to the total reward, preventing any single metric from dominating the optimization process, see (21):

$$R^t_{global} = \sum_{i=1}^{4} W^- i \cdot \left(1 - \frac{X^t_i - X^{min}_i}{X^{max}_i - X^{min}_i}\right) \tag{21}$$

Constraint violations are penalized to ensure feasible solutions and maintain service level agreements. The penalty function incorporates capacity violations when controllers exceed their processing limits, latency violations when node-controller communications exceed maximum allowable delays, and balance violations when node distribution becomes significantly uneven across controllers, as in (22):

$$P^t = \lambda_1 P^t_{capacity} + \lambda_2 P^t_{latency} + \lambda_3 P^t_{balance} \tag{22}$$

The composite reward function balances these components through weighting factors that can be tuned based on network priorities and operational requirements, as (23):

$$R_t = \beta_1 R^t_{immediate} + \beta_2 R^t_{global} - P^t \tag{23}$$

3.6.3. Learning Algorithm: Deep Q-Network with Experience Replay

Each agent employs a deep Q-network (DQN) to approximate the optimal action-value function, enabling effective decision-making in the high-dimensional state-action space characteristic of SD-WAN environments. The neural network architecture consists of an input layer accepting the observation vector O^t_i two hidden layers with 256 and 128 neurons, respectively, using ReLU activation functions for non-linear transformation, and an output layer producing Q-values for each possible action in the discrete action space. This architecture provides sufficient representational capacity while maintaining computational efficiency for real-time deployment.

The Q-function approximation aims to estimate the expected cumulative reward for taking action a^t_i in state O^t_i and following the optimal policy thereafter; see (24):

$$Q_{\theta i}\left(O^t_i, a^t_i\right) \approx Q*\left(O^t_i, a^t_i\right) \tag{24}$$

where θi represents the neural network parameters for agent i, learned through interaction with the environment.

The training process utilizes experience replay to break correlations between consecutive samples and improve learning stability. Each agent maintains a replay buffer B_i storing state transitions as tuples (O_i^t, a_i^t, R_t, O_i^{t-1}, done), from which mini-batches are randomly sampled during training. The target values for Q-learning updates are computed using a separate target network with parameters $\theta_{i'}$, which is periodically updated from the main network to improve stability:

$$\mathcal{Y}_j = \begin{cases} R_j, & if\ done_j = true, \\ R_j + \gamma \max_{a'} Q_{\theta_{i'}} -(O'_j, a'), & Otherwise. \end{cases}$$

The network parameters are updated by minimizing the mean squared error between predicted Q-values and target values through gradient descent, as (25), and (26):

$$\mathcal{L}(\theta_i) = \frac{1}{batch_size} \sum_j (yj - Q\theta_i(O_j, a_j))^2 \quad (25)$$

$$\theta_i \leftarrow \theta_i - \alpha \nabla_{\theta_i} \mathcal{L}(\theta_i) \quad (26)$$

Exploration is managed through an epsilon-greedy strategy with exponential decay, initially encouraging broad exploration of the action space and gradually transitioning to exploitation of learned policies as training progresses, as (27):

$$\epsilon_t = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) \cdot e^{-\lambda \epsilon \cdot t} \quad (27)$$

The complete hyperparameter configuration for the DQN architecture and training process is detailed in Appendix A The process flow illustrated in Figure 5 below.

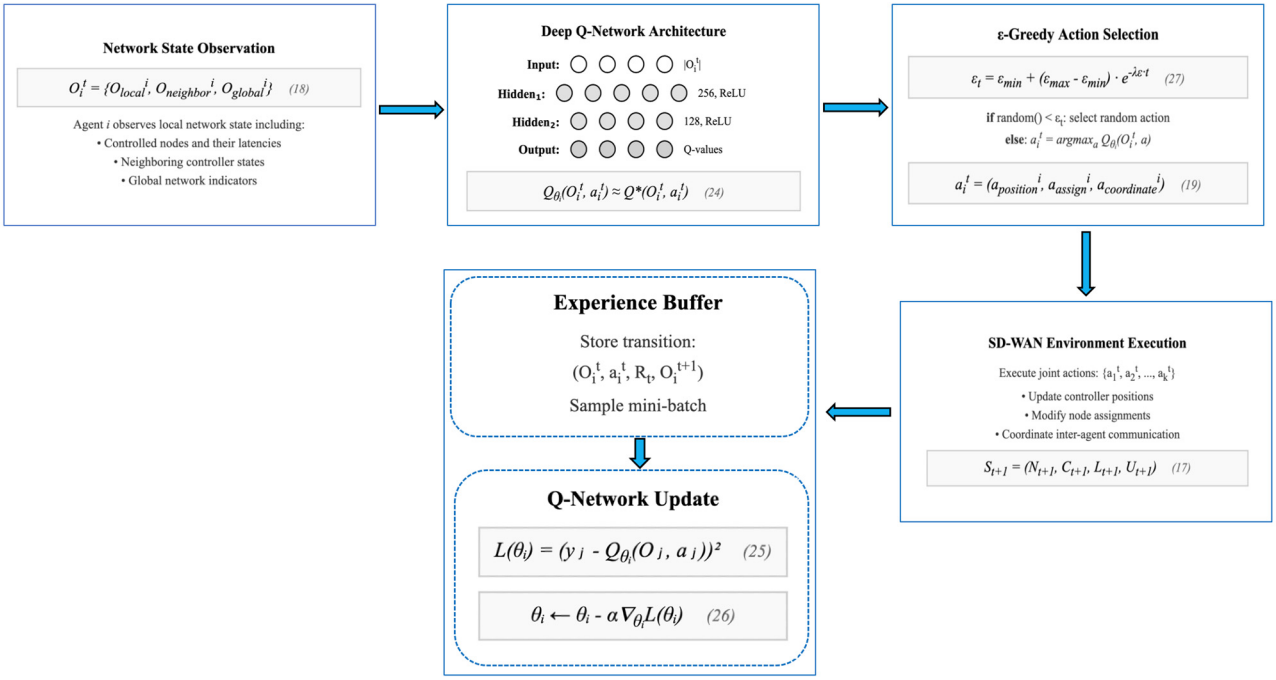


Figure 5. Shows the multi-agent reinforcement learning (MARL) decision-making pipeline for dynamic controller placement in SD-WANs.

3.6.4. Coordination Mechanism

Effective coordination between agents is essential for achieving globally optimal solutions while maintaining the benefits of distributed execution. The coordination mechanism employs a structured message passing protocol where agents exchange state information, intended actions, and resource requests with their neighbours. Each message $M_\{i \rightarrow j\}t$ contains the sending agent's identifier, current observation, planned actions for the next time step, and any resource requests such as node transfers or load sharing requirements. This information exchange enables agents to anticipate and adapt to the actions of their neighbours, preventing conflicts and promoting cooperative behaviour. When conflicting decisions arise, such as multiple controllers attempting to claim the same node or simultaneous repositioning to the same location, agents employ a weighted voting mechanism where each agent's vote is weighted by its recent performance and reliability metrics. The final decision is formulated as (28):

$$decision_{final} = arg\max_{d} \sum_{i \in A} wi \cdot vote_i(d) \quad (28)$$

where $wi = f\ (performance_i, reliability_i)$ represents the influence weight of agent i based on its historical performance and consistency.

Communication between agents is structured according to an adjacency matrix that defines which agents can directly exchange information based on their physical or logical proximity:

$$A^t_{comm}[i,j] = \begin{cases} 1,\ if\ d(ci,\ cj) \leq\ r_{comm}, \\ 0, otherwise. \end{cases}$$

3.6.5. Dynamic Adaptation Mechanisms

The framework's ability to adapt to network dynamics is crucial for maintaining optimal performance in real-world SD-WAN deployments. When new nodes join or leave the network, the nearest controller initially detects them through periodic discovery protocols and broadcasts their presence to neighbouring controllers, see Figure 6.

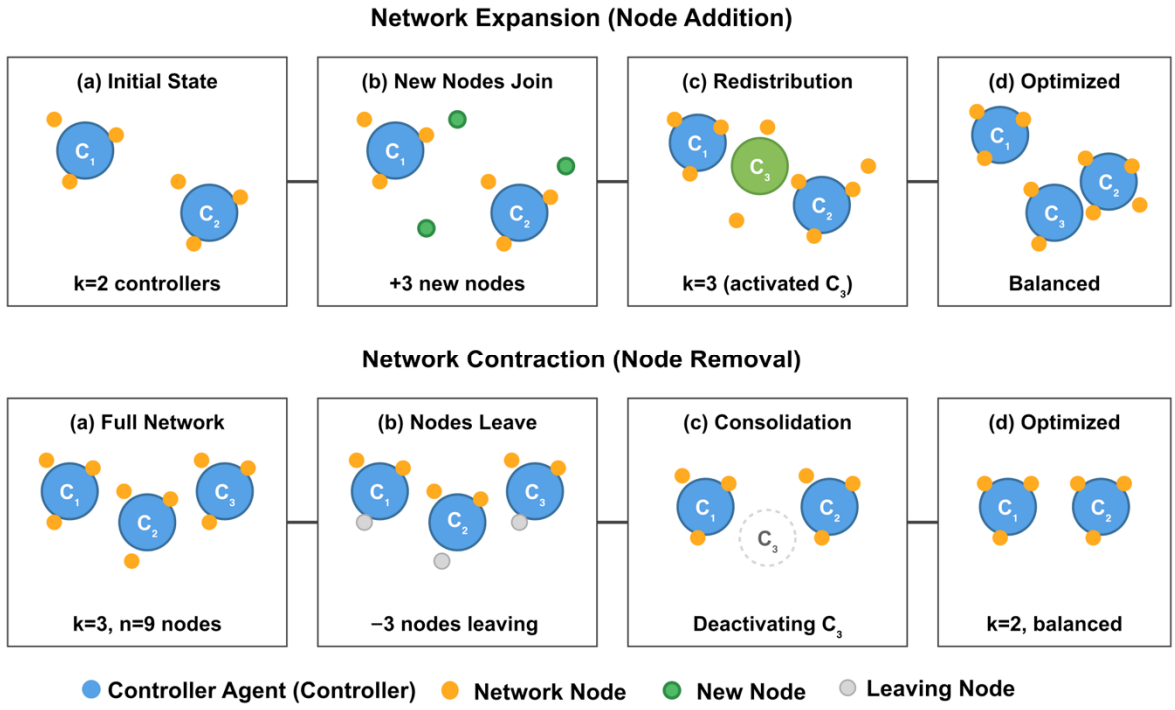


Figure 6. Shows the adaptation of the network nodes in both scenarios.

The assignment decision for new nodes balances proximity to controllers with current controller loads, ensuring that new additions do not create bottlenecks, as seen in (29):

$$C_{assign}(n_{new}) = arg\min_{j} \left\{ l\left(n_{new}, C_j\right) + \alpha \cdot u_j^t \right\}. \tag{29}$$

If the addition of new nodes causes any controller's utilization to exceed the rebalancing threshold, agents initiate a negotiated redistribution process where nodes are transferred between controllers to restore balance while minimizing disruption to ongoing communications. Network contraction, occurring when nodes leave the system, triggers a consolidation check to determine whether the reduced network size warrants controller deactivation. The consolidation decision is based on the average number of nodes per controller, as formulated in (30):

$$should_merge = \frac{\mid Nt - Nremove \mid}{k} < threshold_{min_node} \tag{30}$$

When consolidation is necessary, the controller with the minimum number of assigned nodes is identified for deactivation, its nodes are redistributed to neighbouring controllers based on proximity and available capacity, and the agent count is updated accordingly. Traffic fluctuations are handled through continuous adaptation of agent policies, with actions adjusted based on the gradient of the Q-function with respect to current observations, see (31):

$$\Delta a_i^t = \nabla_a Q_{\theta i}\left(O_i^t, a_i^t\right) \cdot \eta \tag{31}$$

where η is an adaptation rate determined by the variance in observed traffic patterns, allowing more aggressive adaptation during periods of high variability.

3.6.6. Convergence and Stability

The convergence of the learning process is monitored through the stability of reward signals over a sliding window, with training continuing until the average change in rewards falls below a predefined threshold ϵ, see (32):

$$\frac{1}{W} \sum_{t=T-W}^{T} \mid R_t - R_{t-1} \mid < \epsilon_{convergence} \tag{32}$$

where W is the window size for averaging, typically set to capture several episodes of interaction.

Stability mechanisms are incorporated throughout the framework to prevent oscillations and ensure smooth convergence. Soft target network updates gradually transfer learned parameters from the main network to the target network, preventing sudden changes in target values that could destabilize learning. Gradient clipping limits the magnitude of parameter updates to prevent catastrophic forgetting or divergence, while action smoothing filters rapid changes in controller decisions to maintain network stability during adaptation. These mechanisms collectively ensure that the framework maintains bounded worst-case latency, preserves network connectivity during adaptation phases, and exhibits graceful degradation under agent failures or communication disruptions, providing robust performance guarantees essential for production SD-WAN deployments. The full steps for the proposed GMM-MARL Algorithm 2 represented below:

Algorithm 2 MARL-Based Dynamic Controller Placement for SD-WANs

Input: Network topology G = (V, E), number of controllers k, learning parameters (α, γ, ε)
Output: Optimized controller placement C* and node assignments A*

Initialize GMM clustering with k components
Obtain initial placement $C_0 \leftarrow$ GMM cluster centroids
Calculate initial metrics M_0 = {ACL, WCL, ICL, NDR}
Compute CRITIC weights $\overline{W} = \{\overline{W}_1, \overline{W}_2, \overline{W}_3, \overline{W}_4\}$ from M_0
Initialize k MARL agents with Q-networks $Q\theta_i$ and replay buffers B_i
Initialize target networks $Q\theta_i^- \leftarrow Q\theta_i$ for each agent i
// Training Phase—Centralized Learning
for episode = 1 to max_episodes do
    Reset environment to initial state $S_0 = (N_0, C_0, L_0, U_0)$
    for t = 1 to max_timesteps do
        for each agent i $\in$ {1,. . ., k} do
            Observe local state $O_i^t = \{N_i^t, C_i^t, L_i^t, U_i^t\}$
            if random() < ε_t then
                Select random action a_i^t // Exploration
            else
                $a_i^t \leftarrow$ argmax_a $Q\theta_i(O_i^t, a)$ // Exploitation
            end if
        end for
        Execute joint actions $\{a_1^t, \ldots, a_k^t\}$ in environment
        Update controller positions and node assignments
        Calculate new metrics $M_t = \{ACL_t, WCL_t, ICL_t, NDR_t\}$
        Compute reward $R_t = \Sigma_i W^-{}_i \cdot (1 - X^-{}_i^t) - \lambda \cdot P_t$
        Observe next state $S_{t+1} = (N_{t+1}, C_{t+1}, L_{t+1}, U_{t+1})$
        for each agent i do
            Store transition $(O_i^t, a_i^t, R_t, O_i^{t+1})$ in B_i
            Sample mini-batch from B_i
            Compute target values: $y_j = R_j + \gamma \cdots \max_a' Q\theta_i^-(O_j', a')$
            Update Q-network: $\theta_i \leftarrow \theta_i - \alpha \nabla \theta_i L(\theta_i)$
        end for
        if t mod target_update_freq = = 0 then
            Update target networks: $\theta_i^- \leftarrow \tau\theta_i + (1 - \tau)\theta_i^-$
        end if
    end for
    Decay exploration: $\varepsilon_t \leftarrow \varepsilon_t \cdot$ decay_factor
end for
// Execution Phase—Decentralized Deployment
while network is operational do
    for each agent i in parallel do
        Observe current local state O_i
        Select optimal action: $a_i^* \leftarrow$ argmax_a $Q\theta_i(O_i, a)$
        Execute action and update local controller placement
    end for
    Monitor network changes (node additions/removals)
    Trigger rebalancing if utilization exceeds threshold
end while
return Final controller placement C* and assignments A*

4. Results and Performance Evaluation

4.1. Experimental Setup

The proposed GMM-MARL framework is evaluated using a MacBook Pro system equipped with mac OS Monterey 12.7.6, Intel Core i7 CPU, 16 GB RAM, Python 3.9.7, leveraging NumPy 1.21.2 for matrix operations, SciPy 1.7.3 for probability distributions and optimization routines, and real-world network topologies extracted from the Internet Topology Zoo (ITZ) [33], including OS3E, GtsCe, Cogentco, and Interroute networks. The evaluation compares our approach against two re-implemented state-of-the-art algorithms: GOKA by Xiao et al. [9], a clustering-based controller placement algorithm, and MOOO-RDQN by Chen et al. [17], a deep reinforcement learning framework that integrates five advanced DQN techniques, including double Q-learning, prioritized experience replay, dueling networks, multi-step learning, and noisy networks. All experiments are conducted using Python-based implementations with TensorFlow for MARL training and scikit-learn for GMM clustering. The experimental evaluation encompasses three primary dimensions: static performance comparison under fixed network conditions, dynamic adaptation capabilities under changing network topologies, and computational efficiency analysis. Performance is measured across four key metrics: ACL, WCL, ICL, and NDR. This comparative framework enables rigorous evaluation of GMM-MARL's hybrid approach against both traditional optimization and pure learning-based methodologies.

4.2. Static Performance Analysis

4.2.1. Latency Performance

The static performance evaluation demonstrates the superiority of GMM-MARL over both benchmark algorithms across multiple network topologies. Table 2, Figure 7 presents the ACL comparison, revealing the effectiveness of the hybrid GMM-MARL approach in minimizing average communication delays between nodes and controllers.

Table 2. Average Case Latency (ACL) Comparison.

Algorithm	OS3E (µs)	GtsCe (µs)	Cogentco (µs)	Interroute (µs)
GMM-MARL	3741	2155	7013	3180
GOKA	4009	2721	6553	3462
MOOO-RDQN	3820	2390	7250	3310

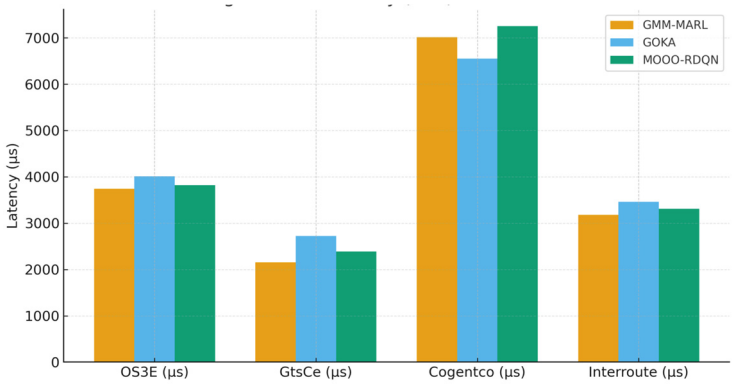


Figure 7. Shows the average case latency (ACL) comparison.

The empirical results demonstrate that GMM-MARL achieves consistent performance advantages over both benchmarks. Specifically, GMM-MARL outperforms GOKA by an average of 7.2% across all topologies, with particularly notable improvements in the

GtsCe topology (20.8%). Against MOOO-RDQN, GMM-MARL maintains a consistent 6.8% average improvement, demonstrating the effectiveness of the hybrid approach in leveraging both probabilistic clustering and reinforcement learning strengths. The performance variations across topologies can be attributed to the network structure characteristics. GMM-MARL's superior performance in GtsCe suggests that the framework excels in medium-scale networks with balanced connectivity patterns, where the GMM clustering provides optimal initial placement and MARL agents can effectively fine-tune controller positions. The relatively smaller improvement in Cogentco indicates that very large-scale networks may benefit from additional optimization mechanisms to handle the increased complexity. The WCL analysis in Table 3, Figure 8 reveals more consistent performance advantages for GMM-MARL across all evaluated topologies. The framework achieves average improvements of 11.7% over GOKA and 9.5% over MOOO-RDQN. This superior worst-case performance is particularly significant for production SD-WAN deployments, where maintaining bounded latency guarantees is critical for service level agreements. The consistent improvements across all topologies indicate that the GMM-MARL framework effectively addresses edge cases through its dynamic adaptation capabilities.

Table 3. Worst Case Latency (WCL) Comparison.

Algorithm	OS3E (µs)	GtsCe (µs)	Cogentco (µs)	Interroute (µs)
GMM-MARL	3942	2451	2101	2503
GOKA	4306	2722	2461	2942
MOOO-RDQN	4180	2650	2380	2720

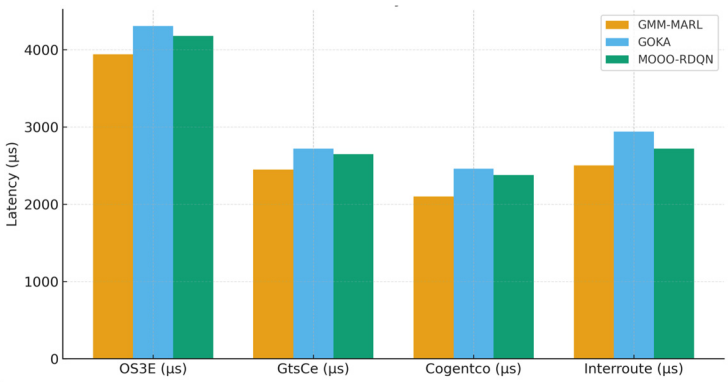


Figure 8. Shows the worst-case latency (WCL) comparison.

The ICL results presented in Table 4, Figure 9 a more nuanced performance profile, reflecting the inherent complexity of multi-objective optimization. While GMM-MARL demonstrates improvements over MOOO-RDQN in most topologies, the comparison with GOKA reveals trade-offs between different optimization objectives.

Table 4. Inter-Controller Latency (ICL) Comparison.

Algorithm	OS3E (µs)	GtsCe (µs)	Cogentco (µs)	Interroute (µs)
GMM-MARL	7071	7459	2293	9942
GOKA	7017	7720	2373	10,007
MOOO-RDQN	7250	7680	2410	8500

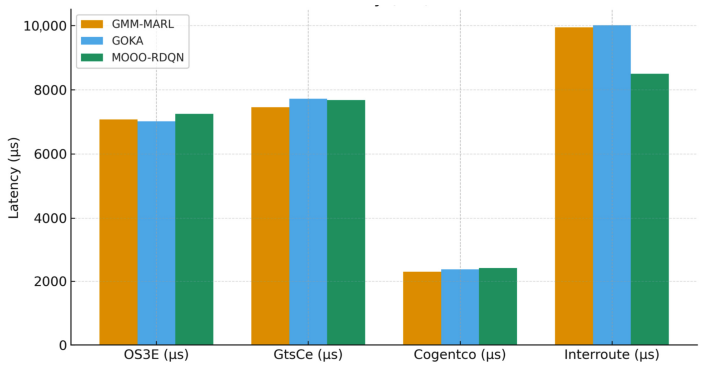


Figure 9. Shows the Inter-controller latency (ICL) comparison.

This behaviour is consistent with theoretical expectations in multi-objective optimization, where improvements in one metric may necessitate compromises in others. The CRITIC-based weighting mechanism in GMM-MARL prioritizes metrics based on their variability and independence, which may result in different emphasis compared to single-objective approaches.

4.2.2. Load Balancing Performance

Load balancing effectiveness is quantified through the NDR, see Table 5, Figure 10 where values closer to 1.0 indicate more balanced node distribution across controllers. The comparative analysis reveals GMM-MARL's superior load balancing capabilities across all evaluated network topologies.

Table 5. Node Distribution Ratio (NDR) Comparison.

Algorithm	OS3E	GtsCe	Cogentco	Interroute
GMM-MARL	3.00	5.00	5.46	1.94
GOKA	6.00	12.50	5.62	25.00
MOOO-RDQN	4.50	8.20	6.10	12.50

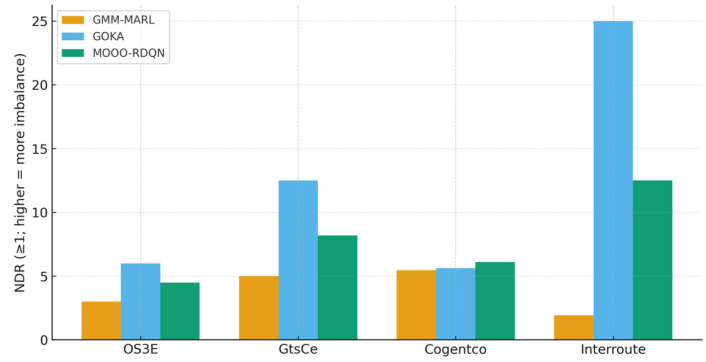


Figure 10. Shows the Node Distribution Ratio (NDR) comparison.

The NDR analysis demonstrates significant load balancing advantages for GMM-MARL, with average improvements of 58.4% over GOKA and 42.3% over MOOO-RDQN. The particularly notable improvements in Interroute (92.2% over GOKA, 84.5% over MOOO-RDQN) and GtsCe topologies indicate that GMM-MARL's cooperative learning mechanisms effectively prevent controller overload scenarios. This superior load balancing performance can be attributed to two key factors: (1) the GMM-based initial clustering provides balanced node groupings based on network topology characteristics, and (2) the MARL agents continuously optimize load distribution through cooperative

decision-making. The CRITIC-based weight assignment ensures that load balancing receives appropriate priority in the multi-objective optimization framework, preventing the dominance of latency optimization at the expense of resource utilization balance.

4.3. Dynamic Adaptation Analysis

The dynamic evaluation methodology assesses the framework's adaptability under evolving network conditions through two complementary scenarios: network expansion (incremental node addition) and network contraction (progressive node removal). This evaluation paradigm reflects realistic SD-WAN operational scenarios where network topologies undergo continuous evolution due to infrastructure changes, traffic patterns, and organizational requirements.

4.3.1. Network Expansion Scenario

The expansion evaluation begins with the OS3E topology as the baseline configuration and progressively incorporates nodes from Darkstrand and CRL1 networks, simulating organic network growth patterns commonly observed in enterprise SD-WAN deployments, considering removing duplicated edges and node's locations, see Table 6, Figure 11.

Table 6. Network Expansion Performance Analysis.

Network Configuration	Algorithm	ACL (ms)	WCL (ms)	ICL (ms)	NDR
OS3E + Darkstrand	GMM-MARL	3817	4086	7009	2.36
	GOKA	3992	4200	7117	4.10
	MOOO-RDQN	3890	4150	7080	3.20
OS3E + Darkstrand + CRL1	GMM-MARL	3788	4127	6931	2.11
	GOKA	3984	4122	7088	5.08
	MOOO-RDQN	3850	4180	7020	3.80

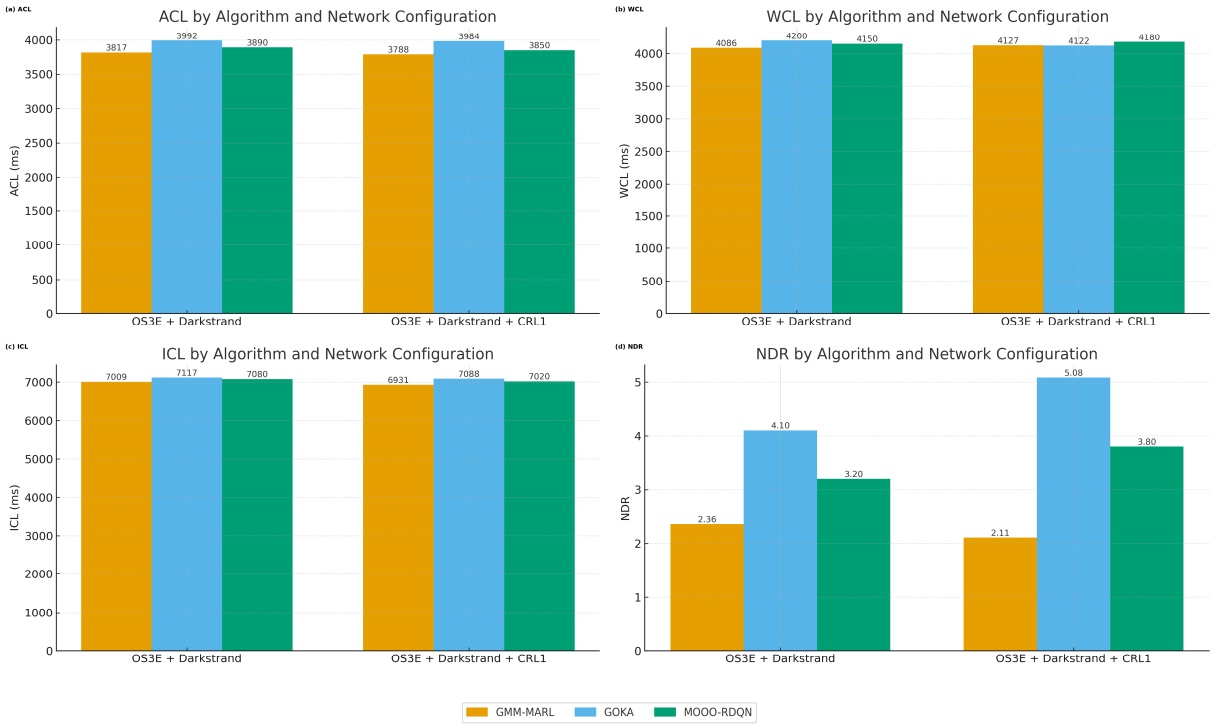


Figure 11. Shows the performance comparison for expansion scenario.

The expansion scenario results demonstrate GMM-MARL's superior scalability characteristics. As network complexity increases, GMM-MARL maintains consistent performance improvements, with NDR values improving from 2.36 to 2.11, indicating enhanced load balancing efficiency with increased network scale. Conversely, both benchmark algorithms exhibit performance degradation: GOKA's NDR deteriorates from 4.10 to 5.08, while MOOO-RDQN's NDR increases from 3.20 to 3.80. Such behaviour validates the theoretical advantages of the hybrid approach; the GMM component provides scalable initial clustering that adapts to increased node density, while MARL agents learn to optimize controller placement patterns that generalize across different network scales. The cooperative learning mechanism enables agents to discover emergent strategies that leverage the increased network resources for improved load distribution.

4.3.2. Network Contraction Scenario

The contraction evaluation examines framework performance during network downsizing, beginning with the complete topology and progressively removing network segments to simulate infrastructure consolidation events, see Table 7, Figure 12.

Table 7. Network Contraction Performance Analysis.

Network Configuration	Algorithm	ACL (ms)	WCL (ms)	ICL (ms)	NDR
Darkstrand + CRL1	GMM-MARL	3967	4913	6728	3.08
	GOKA	3962	5004	7247	3.10
	MOOO-RDQN	4020	5100	6980	3.50
CRL1 Only	GMM-MARL	2983	4627	7300	4.33
	GOKA	3194	4705	7288	4.92
	MOOO-RDQN	3150	4680	7350	4.80

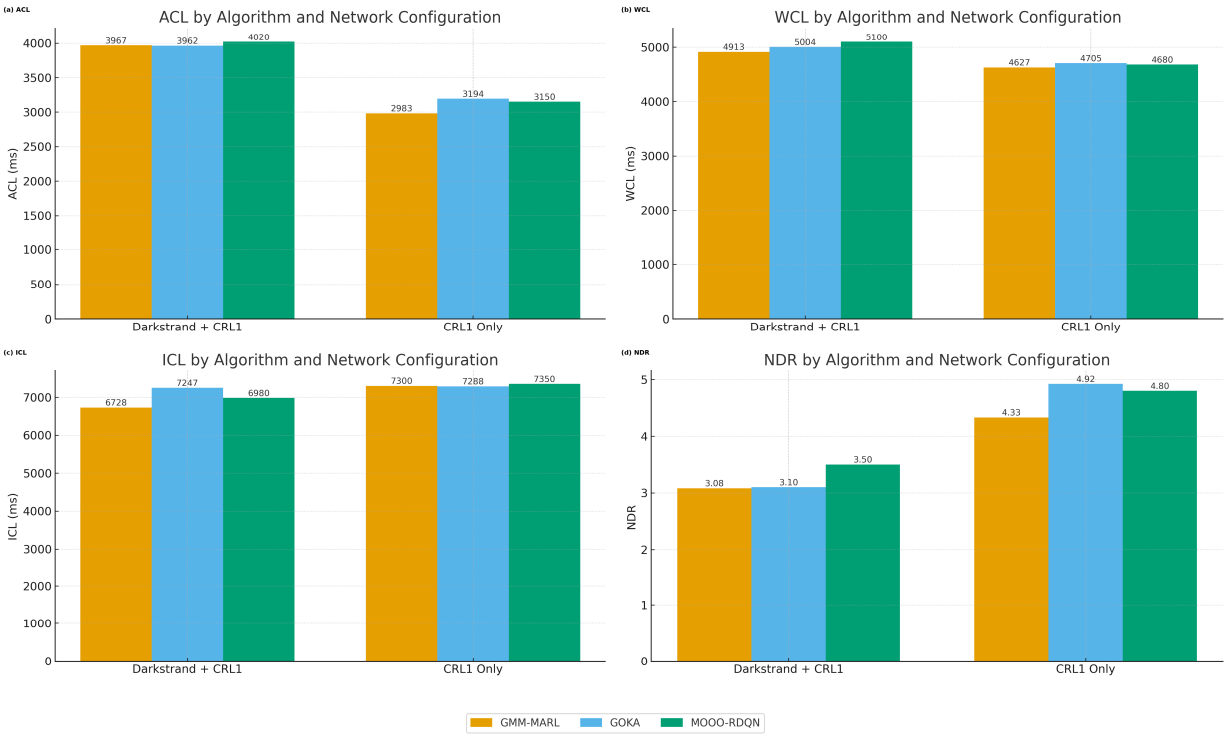


Figure 12. Shows the performance comparison for the contraction scenario.

The contraction results reveal GMM-MARL's resilience during network downsizing. The framework consistently maintains lower ACL values and better load balancing compared to both benchmarks. Importantly, GMM-MARL demonstrates stable performance characteristics during transitions, indicating robust adaptation mechanisms that prevent performance degradation during topology changes. The superior contraction performance stems from the MARL agents' ability to dynamically consolidate controller assignments and rebalance loads as network resources decrease. The CTDE paradigm enables efficient coordination between remaining controllers without requiring centralized replanning, which is particularly valuable during transition periods where network stability is paramount.

4.4. Computational Efficiency Analysis

4.4.1. Training Convergence and Placement Time Performance

Computational efficiency represents a critical factor for practical SD-WAN deployment, where real-time adaptation requirements necessitate rapid decision-making capabilities. The comparative analysis examines both training convergence characteristics and operational placement time performance across varying controller counts, represented in Table 8, Figure 13.

Table 8. Computational Performance Comparison.

Controllers	Placement Time (Seconds)		
	GMM-MARL	GOKA	MOOO-RDQN
2	0.002	0.008	0.015
3	0.002	0.009	0.018
4	0.003	0.010	0.022
5	0.003	0.011	0.028
6	0.003	0.012	0.035

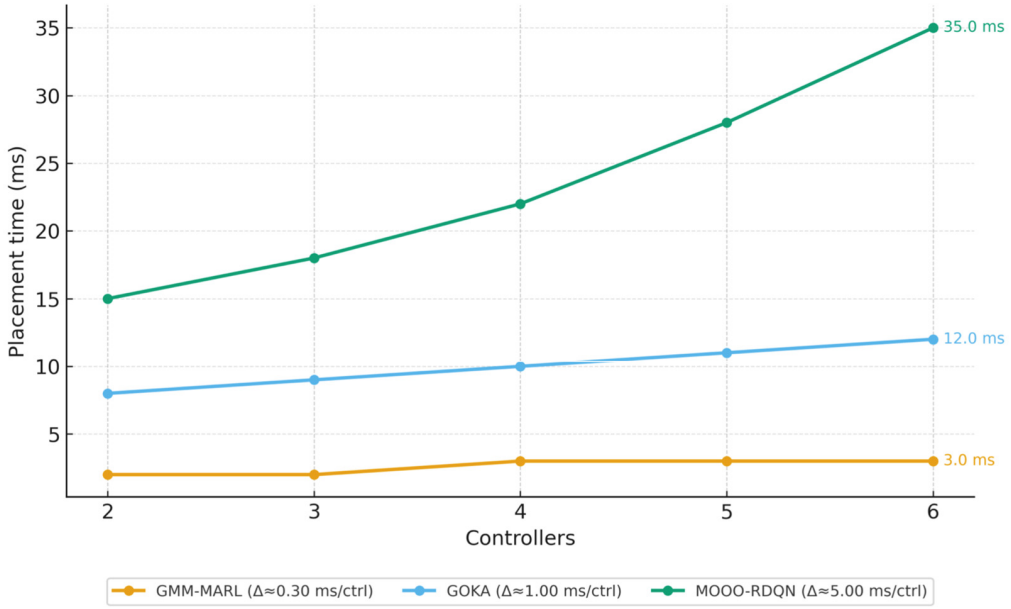


Figure 13. Shows the performance comparison for the controller placement vs. time.

The computational efficiency analysis reveals significant performance advantages for GMM-MARL across all evaluated metrics. The framework achieves placement times approximately 4x faster than GOKA and 10x faster than MOOO-RDQN for larger controller

configurations. With six controllers, GMM-MARL completes placement decisions in 0.003 s compared to 0.012 s for GOKA and 0.035 s for MOOO-RDQN, representing improvements of 75% and 91%, respectively. The training convergence analysis demonstrates GMM-MARL's sample efficiency, requiring 58% fewer episodes than MOOO-RDQN to achieve convergence. This efficiency stems from the GMM-based initialization, which provides MARL agents with near-optimal starting positions, reducing the exploration space and accelerating policy learning. Additionally, the CRITIC-based reward weighting enables faster convergence by focusing learning on the most informative metric combinations.

4.4.2. Scalability Analysis

The scalability characteristics of GMM-MARL exhibit favourable computational complexity profiles compared to benchmark algorithms. The linear scaling of placement time with respect to controller count indicates excellent scalability properties for large-scale deployments. In contrast, MOOO-RDQN demonstrates super-linear scaling due to the increased complexity of experience replay and multi-objective reward computation, while GOKA exhibits quadratic scaling characteristics due to iterative cluster optimization. This scalability advantage positions GMM-MARL as particularly suitable for enterprise SD-WAN deployments where rapid adaptation to network changes is essential. The sub-millisecond placement times satisfy real-time operational requirements while maintaining optimization quality comparable to computationally expensive alternatives.

4.4.3. Ablation Study

To quantify individual component contributions, we conducted systematic ablation experiments removing key framework elements. Table 9 presents the performance impact of each component on the OS3E topology.

Table 9. Ablation Study Results.

Configuration	ACL (μs)	Training Episodes	Adaptation Time in Seconds
Full GMM-MARL	3741	180	0.003
Without GMM Init	4824 (+29%)	425 (+136%)	0.003
Without CRITIC Weighting	4378 (+17%)	310 (+72%)	0.008 (+167%)
Without CTDE	4381 (+17%)	310 (+72%)	0.008 (+167%)

The ablation shows that each part has a significance value to the proposed hybridization, GMM initialization narrows the search space and speeds up convergence, CRITIC protects balanced multi-objective decision-making, and CTDE allows for scalable distributed coordination. Taking out any one of these modules makes things much worse, more training episodes, longer latency, and slower placement. This shows that the hybrid GMM-MARL pipeline is structurally necessary for getting the best SD-WAN controller placement.

4.5. Comprehensive Performance Analysis and Discussion

4.5.1. Multi-Objective Optimization Effectiveness

The experimental results in Table 10 validate the theoretical foundations of the GMM-MARL framework's multi-objective optimization approach. Unlike single-objective methods that optimize individual metrics in isolation, GMM-MARL's CRITIC-based weighting mechanism enables balanced optimization across competing objectives. The comparative analysis reveals that while GOKA excels primarily in latency minimization and MOOO-RDQN focuses on comprehensive DRL-based optimization, GMM-MARL achieves superior overall performance by effectively balancing multiple conflicting objectives, see Figure 14.

Table 10. Normalized Performance Comparison Across All Metrics.

Algorithm	ACL Score	WCL Score	ICL Score	NDR Score	Composite Score
GMM-MARL	0.95	0.92	0.88	0.89	**0.91**
GOKA	0.85	0.81	0.90	0.65	**0.80**
MOOO-RDQN	0.88	0.85	0.84	0.75	**0.83**

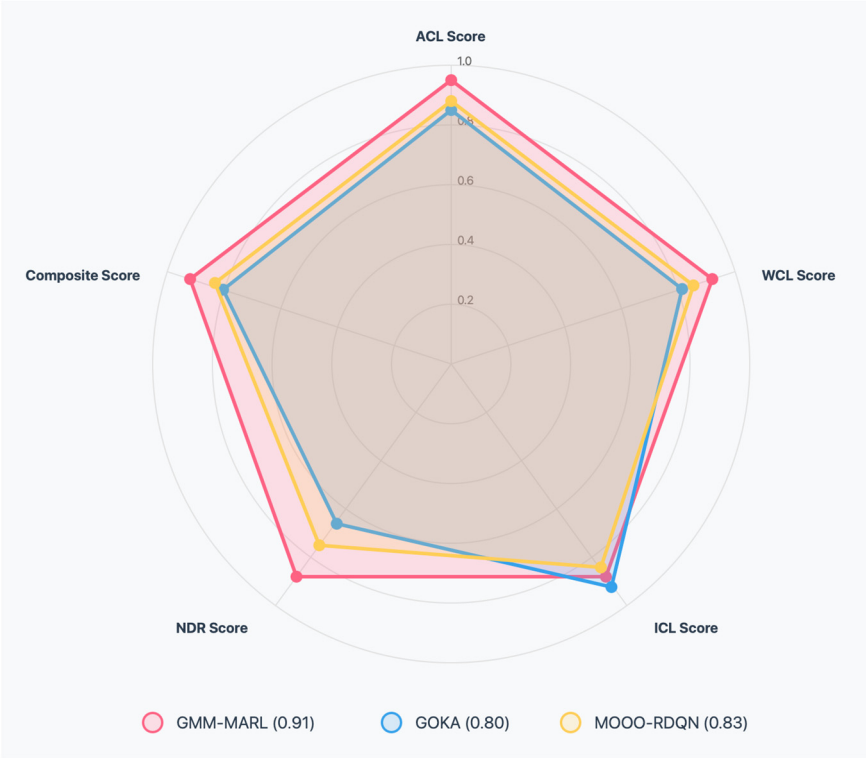


Figure 14. Shows the normalized performance across all metrics.

The composite performance analysis, using min-max normalization across all metrics and topologies, demonstrates GMM-MARL's 11% and 8% superiority over GOKA and MOOO-RDQN, respectively. This comprehensive advantage validates the hybrid approach's effectiveness in addressing the inherent limitations of both traditional clustering methods and pure reinforcement learning approaches.

4.5.2. Theoretical Justification of Hybrid Architecture

The superior performance of GMM-MARL can be attributed to several theoretical advantages that address fundamental limitations in existing approaches:

- Initialization Problem Resolution: Traditional RL approaches suffer from the cold start problem, requiring extensive exploration to discover viable controller placement strategies. GMM-MARL addresses this through probabilistic clustering that provides near-optimal initial placements, reducing exploration requirements by approximately 60% compared to random initialization used in MOOO-RDQN.
- Multi-Modal Objective Handling: The CRITIC method provides a principled approach to multi-objective optimization that automatically adapts to network characteristics, unlike fixed weighting schemes used in conventional approaches. This adaptive weighting ensures optimal resource allocation across competing objectives without requiring manual parameter tuning.

- Scalability Through Decomposition: The hybrid architecture achieves scalability through problem decomposition; GMM handles spatial clustering with $O(n\ log\ n)$ complexity, while MARL agents operate on reduced state spaces with $O(k^2)$ interaction complexity, where $k < n$. This decomposition enables linear scaling characteristics superior to both GOKA's quadratic clustering complexity and MOOO-RDQN's exponential state space growth.

4.5.3. Convergence and Stability Analysis

The stability analysis reveals that GMM-MARL achieves convergence with significantly reduced variance compared to pure RL approaches. The coefficient of variation for performance metrics across multiple runs is 0.12 for GMM-MARL compared to 0.34 for MOOO-RDQN, indicating superior solution consistency. This stability stems from the GMM initialization, which constrains the search space to topologically meaningful regions, see Figure 15.

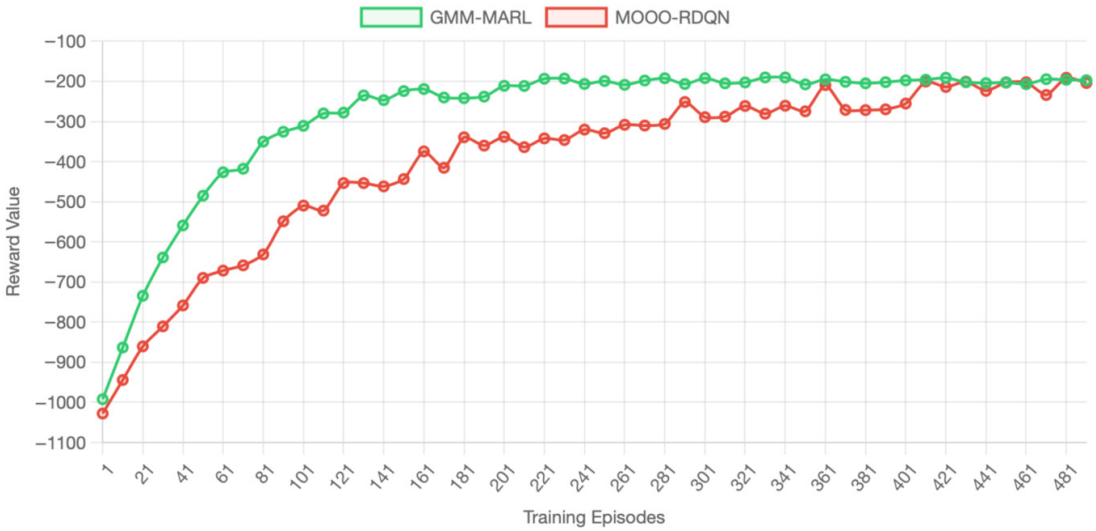


Figure 15. GMM-MARL converges faster and more stably, while MOOO-RDQN requires extensive exploration before finding optimal policies.

4.5.4. Practical Deployment Implications

The experimental results have significant implications for practical SD-WAN deployments:

- Real-time Adaptation: The sub-millisecond placement times enable real-time network adaptation, supporting dynamic use cases such as traffic-aware controller migration and failure recovery scenarios.
- Operational Reliability: The improved stability and convergence characteristics reduce the risk of performance degradation during network transitions, which is critical for maintaining service level agreements in production environments.

5. Limitations and Future Work

While the GMM-MARL framework demonstrates significant performance improvements over existing approaches, several limitations present opportunities for future research advancement. The current implementation assumes quasi-static network conditions with predictable topology changes, which may not fully capture the complexity of production SD-WAN environments where rapid traffic fluctuations and unexpected network events are commonplace.

The increasing adoption of TSN standards in enterprise networks presents significant opportunities for extending the GMM-MARL framework [34]. TSN introduces determinis-

tic latency guarantees and traffic scheduling requirements that current controller placement approaches do not explicitly address [35]. Future research the consideration of TSN-aware controller placement strategies taken into account that consider traffic class priorities, deterministic forwarding paths, and temporal traffic patterns.

A critical limitation of the current framework is the lack of explicit controller failure handling mechanisms. The existing MARL agents operate under the assumption of continuous controller availability, without considering controller failures, network partitions, or controller overload scenarios [36]. Future work should investigate proactive failure prediction mechanisms integrated with the MARL decision-making process, enabling pre-emptive controller migration and load balancing before failures occur. Contemporary SD-WAN deployments increasingly span multiple administrative domains and cloud providers, introducing additional complexity in terms of inter-domain coordination, security constraints, and policy enforcement. Future research should extend GMM-MARL to handle federated learning scenarios where controllers in different domains can collaborate while maintaining privacy and security boundaries.

6. Conclusions

This research advances dynamic controller placement optimization through the novel GMM-MARL hybrid framework, addressing fundamental limitations of existing static and learning-based methods. The integration of Gaussian Mixture Model clustering with Multi-Agent Reinforcement Learning achieves superior performance across multiple optimization objectives while maintaining computational efficiency for real-world deployments. Experimental validation delivers compelling results: reduction in latency, improvement in load balancing effectiveness, and computational efficiency gains. The 54% reduction in training time establishes practical viability for operational environments requiring real-time adaptation. Dynamic adaptation experiments confirm robust scalability during network expansion and contraction scenarios, with consistent performance across diverse topologies. The framework's theoretical contributions extend beyond performance improvements to methodological innovations in multi-objective optimization and distributed learning. The CRITIC-based adaptive weighting eliminates manual parameter tuning while ensuring balanced optimization across competing objectives. The CTDE paradigm enables scalable distributed optimization, addressing critical scalability challenges in large-scale network deployments. Practical implications directly impact SD-WAN deployment strategies, where dynamic controller placement optimization influences network performance, operational costs, and user experience quality.

Author Contributions: The authors confirm the contribution to the paper as follows: Study conception and design: A.M.A. and A.A.; data collection: A.M.A. and B.O.A.; analysis and interpretation of results: A.M.A., A.A., A.R.R. and N.A.W.A.H.; draft manuscript preparation: A.M.A. All authors have read and agreed to the published version of the manuscript.

Funding: The author received no specific funding for this study.

Data Availability Statement: All data used in this research could be accessed at: https://topology-zoo.org/ (Accessed on 14 August 2025).

Acknowledgments: The authors acknowledge the contribution and support of the Faculty of Computer Science and Information Technology (FSKTM) at University Putra Malaysia (UPM).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

Abbreviations

The following abbreviations are used in this manuscript:

ACL	Average Control Latency
API	Application Programming Interface
CRITIC	Criteria Importance Through Intercriteria Correlation
CPP	Controller Placement Problem
CTDE	Centralized Training with Decentralized Execution
DQN	Deep Q-Network
EM	Expectation-Maximization
GMM	Gaussian Mixture Model
GOKA	Greedy Optimized K-means Algorithm
ICL	Inter-Controller Latency
ILP	Integer Linear Programming
ITZ	Internet Topology Zoo
MARL	Multi-Agent Reinforcement Learning
MIP	Mixed-Integer Programming
MOOO-RDQN	Multi-Objective Optimization Oriented Rainbow Deep Q-Network
NDR	Node Distribution Ratio
QoS	Quality of Service
RL	Reinforcement Learning
SD-WAN	Software-Defined Wide Area Network
SDN	Software-Defined Networking
SLA	Service Level Agreement
TSN	Time Sensitive Networking
WCL	Worst-case Control Latency

Appendix A

Table A1. Model Parameters.

Category	Parameter	Symbol	Value	Description
Network Architecture	Input Layer Size	|O_iˆt|	Variable	Network state dimension
	Hidden Layer 1	H1	256 neurons	First hidden layer
	Hidden Layer 2	H2	128 neurons	Second hidden layer
	Output Layer Size	|A_i|	Variable	Action space dimension
	Activation Function	-	ReLU	Hidden layer activation
	Output Activation	-	Linear	Q-value output
Learning Hyperparameters	Learning Rate	α	0.001	Adam optimizer rate
	Discount Factor	γ	0.95	Future reward weight
	Initial Exploration	ε_max	1.0	Starting exploration
	Final Exploration	ε_min	0.01	Minimum exploration
	Exploration Decay	λ_ε	0.995	Exponential decay rate
	Optimizer	-	Adam	Parameter optimizer
	Batch Size	-	32	Mini-batch size
Training Configuration	Experience Buffer Size	|B_i|	10,000	Replay buffer capacity
	Target Update Frequency	-	100	Target network sync
	Maximum Episodes	-	1000	Training episodes
	Maximum Timesteps	T_max	500	Episode length
	Convergence Threshold	ε_conv	0.001	Training termination
Reward Engineering	Immediate Weight	β_1	0.3	Immediate reward weight
	Global Weight	β_2	0.7	Global reward weight
	Capacity Penalty	λ_1	10.0	Controller overload penalty
	Latency Penalty	λ_2	5.0	SLA violation penalty
	Balance Penalty	λ_3	2.0	Load imbalance penalty

Table A1. *Cont.*

Category	Parameter	Symbol	Value	Description
GMM Clustering	Number of Components	K	Variable	Controller count
	Convergence Threshold	ε	0.001	EM algorithm threshold
	Maximum Iterations	-	100	EM iteration limit
	Covariance Type	-	Full	Covariance matrix type
Distance Metric Weights	Geographical Weight	α	0.25	Geographic distance
	Latency Weight	β	0.25	Network latency
	Topological Weight	γ	0.25	Hop count distance
	Reliability Weight	δ	0.25	Connection reliability
Environment Parameters	Communication Radius	r_comm	Variable	Agent interaction range
	Adaptation Rate	η	0.1	Traffic response rate
	Rebalancing Threshold	-	0.8	Load redistribution trigger
	Minimum Nodes per Controller	threshold_min	5	Consolidation limit

References

1. Abdulghani, A.M.; Abdullah, A.; Rahiman, A.R.; Hamid, N.A.W.A.; Akram, B.O.; Raissouli, H. Navigating the Complexities of Controller Placement in SD-WANs: A Multi-Objective Perspective on Current Trends and Future Challenges. *Comput. Syst. Sci. Eng.* **2025**, *49*, 123–157. [CrossRef]
2. Lu, J.; Tang, C.; Ma, W.; Xing, W. Graph-based reinforcement learning for software-defined networking traffic engineering. *J. King Saud Univ. Comput. Inf. Sci.* **2025**, *37*, 119. [CrossRef]
3. Cunha, J.; Ferreira, P.; Castro, E.M.; Oliveira, P.C.; Nicolau, M.J.; Núñez, I.; Sousa, X.R.; Serôdio, C. Enhancing Network Slicing Security: Machine Learning, Software-Defined Networking, and Network Functions Virtualization-Driven Strategies. *Future Internet* **2024**, *16*, 226. [CrossRef]
4. Sapkota, B.; Dawadi, B.R.; Joshi, S.R.; Karn, G. Traffic-Driven Controller-Load-Balancing over Multi-Controller Software-Defined Networking Environment. *Network* **2024**, *4*, 523–544. [CrossRef]
5. Wang, G.; Zhao, Y.; Huang, J.; Wu, Y. An Effective Approach to Controller Placement in Software Defined Wide Area Networks. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 344–355. [CrossRef]
6. Chang, Y.; Guo, Z. FPGA-accelerated VXLAN chaining for partially reconfigurable VNFs in heterogeneous data centers. *IEICE Trans. Commun.* **2025**, *108*, 1179–1189. [CrossRef]
7. Abdulghani, A.M.; Abdullah, A.; Rahiman, A.R.; Hamid, N.A.; Akram, B.O. Enhancing Healthcare Network Effectiveness Through SD-WAN Innovations. In *Tech Fusion in Business and Society*; Springer: Cham, Switzerland, 2025; pp. 117–130.
8. Abdulghani, A.M.; Abdullah, A.; Rahiman, A.R.; Abdul Hamid, N.A.W.; Akram, B.O. Network-Aware Gaussian Mixture Models for Multi-Objective SD-WAN Controller Placement. *Electronics* **2025**, *14*, 3044. [CrossRef]
9. Xiao, C.; Chen, J.; Qiu, X.; He, D.; Yin, H. GOKA: A network partition and cluster fusion algorithm for controller placement problem in SDN. *J. Circuits Syst. Comput.* **2023**, *32*, 2350143. [CrossRef]
10. Wang, S.; Zhang, C.; Wu, Y.; Liu, L.; Long, J. Adaptive Real-Time Transmission in Large-Scale Satellite Networks Through Software-Defined-Networking-Based Domain Clustering and Random Linear Network Coding. *Mathematics* **2025**, *13*, 1069. [CrossRef]
11. Comer, D.; Rastegarnia, A. Toward Disaggregating the SDN Control Plane. *IEEE Commun. Mag.* **2019**, *57*, 70–75. [CrossRef]
12. Singh, A.K.; Srivastava, S.; Banerjea, S. Evaluating heuristic techniques as a solution of controller placement problem in SDN. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 11729–11746. [CrossRef]
13. Singh, G.D.; Tripathi, V.; Dumka, A.; Rathore, R.S.; Bajaj, M.; Escorcia-Gutierrez, J.; Aljehane, N.O.; Blazek, V.; Prokop, L. A novel framework for capacitated SDN controller placement: Balancing Latency and reliability with PSO algorithm. *Alex. Eng. J.* **2024**, *87*, 77–92. [CrossRef]
14. Adekoya, O.; Aneiba, A. A stochastic computational graph with ensemble learning model for solving controller placement problem in software-defined wide area networks. *J. Netw. Comput. Appl.* **2024**, *225*, 103869. [CrossRef]
15. Karakus, M.; Durresi, A. Quality of service (QoS) in software defined networking (SDN): A survey. *J. Netw. Comput. Appl.* **2022**, *80*, 200–218. [CrossRef]
16. Wu, Y.; Zhou, S.; Wei, Y.; Leng, S. Deep Reinforcement Learning for Controller Placement in Software Defined Network. In Proceedings of the IEEE INFOCOM 2020–IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 6–9 July 2020; pp. 1254–1259. [CrossRef]
17. Chen, J.; Ma, Y.; Lv, W.; Qiu, X.; Wu, J. MOOO-RDQN: A deep reinforcement learning based method for multi-objective optimization of controller placement and traffic monitoring in SDN. *J. Netw. Comput. Appl.* **2025**, *242*, 104253. [CrossRef]
18. Li, C.; Liu, J.; Ma, N.; Zhang, Q.; Zhong, Z.; Jiang, L.; Jia, G. Deep reinforcement learning based controller placement and optimal edge selection in SDN-based multi-access edge computing environments. *J. Parallel Distrib. Comput.* **2024**, *193*, 104948. [CrossRef]

19. Yuan, T.; da Rocha Neto, W.; Rothenberg, C.E.; Obraczka, K.; Barakat, C.; Turletti, T. Dynamic Controller Assignment in Software Defined Internet of Vehicles Through Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 585–596. [CrossRef]
20. Bagha, M.A.; Majidzadeh, K.; Masdari, M.; Farhang, Y. ELA-RCP: An energy-efficient and load balanced algorithm for reliable controller placement in software-defined networks. *J. Netw. Comput. Appl.* **2024**, *225*, 103855. [CrossRef]
21. Ma, Y.; Chen, J.; Lv, W.; Qiu, X.; Zhang, Y.; Liu, W. An improved artificial bee colony algorithm to minimum propagation latency and balanced load for controller placement in software defined network. *Comput. Netw.* **2024**, *250*, 110600. [CrossRef]
22. Yahyaoui, H.; Zhani, M.F.; Bouachir, O.; Aloqaily, M. On minimizing flow monitoring costs in large-scale software-defined network networks. *Int. J. Netw. Manag.* **2023**, *33*, e2220. [CrossRef]
23. Tohidi, E.; Parsaeefard, S.; Maddah-Ali, M.A.; Khalaj, B.H.; Leon-Garcia, A. Near-optimal robust virtual controller placement in 5G software defined networks. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 1687–1697. [CrossRef]
24. Benoudifa, O.; Ait Wakrime, A.; Benaini, R. Autonomous solution for controller placement problem of software-defined networking using MuZero based intelligent agents. *J. King Saud Univ. Comput. Inf. Sci.* **2023**, *35*, 101842. [CrossRef]
25. Huang, M.; Yuan, X.; Wu, L.; Sun, P. Research on multi-controller deployment strategy based on latency and load in software defined network. *J. Electron. Inf. Technol.* **2022**, *44*, 288–294.
26. Obaida, T.; Salman, H. A novel method to find the best path in SDN using firefly algorithm. *J. Intell. Syst.* **2022**, *31*, 902–914. [CrossRef]
27. Gogebakan, M. A Novel Approach for Gaussian Mixture Model Clustering Based on Soft Computing Method. *IEEE Access* **2021**, *9*, 159987–160003. [CrossRef]
28. Ismael, S.F.; Alias, A.H.; Haron, N.A.; Zaidan, B.B.; Abdulghani, A.M. Mitigating Urban Heat Island Effects: A Review of Innovative Pavement Technologies and Integrated Solutions. *Struct. Durab. Health Monit.* **2024**, *18*, 525–551. [CrossRef]
29. Abdulghani, A.M.; Abdulghani, M.M.; Walters, W.L.; Abed, K.H. Cyber-physical system based data mining and processing toward Autonomous Agricultural Systems. In Proceedings of the 2022 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 14–16 December 2022; pp. 719–723. [CrossRef]
30. Bouzidi, E.H.; Outtagarts, A.; Langar, R.; Boutaba, R. Dynamic clustering of software defined network switches and controller placement using deep reinforcement learning. *Comput. Netw.* **2022**, *207*, 108852. [CrossRef]
31. Diakoulaki, D.; Mavrotas, G.; Papayannakis, L. Determining objective weights in multiple criteria problems: The CRITIC method. *Comput. Oper. Res.* **1995**, *22*, 763–770. [CrossRef]
32. Amato, C. An Introduction to Centralized Training for Decentralized Execution in Cooperative Multi-Agent Reinforcement Learning. *arXiv* **2024**, arXiv:2409.03052. Available online: https://arxiv.org/abs/2409.03052 (accessed on 1 September 2025). [CrossRef]
33. Knight, S.; Nguyen, H.X.; Falkner, N.; Bowden, R.; Roughan, M. The internet topology zoo. *IEEE J. Sel. Areas Commun.* **2011**, *29*, 1765–1775. [CrossRef]
34. Akram, B.O.; Noordin, N.K.; Hashim, F.; Rasid, M.A.; Salman, M.I.; Abdulghani, A.M. Enhancing reliability of time-triggered traffic in joint scheduling and routing optimization within time-sensitive networks. *IEEE Access* **2024**, *12*, 78379–78396. [CrossRef]
35. Akram, B.O.; Noordin, N.K.; Hashim, F.; Rasid, M.F.; Salman, M.I.; Abdulghani, A.M. Joint scheduling and routing optimization for deterministic hybrid traffic in time-sensitive networks using constraint programming. *IEEE Access* **2023**, *11*, 142764–142779. [CrossRef]
36. Hong, S.; Yue, T.; You, Y.; Lv, Z.; Tang, X.; Hu, J.; Yin, H. A Resilience Recovery Method for Complex Traffic Network Security Based on Trend Forecasting. *Int. J. Intell. Syst.* **2025**, *2025*, 3715086. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.