BAREKENG: Journal of Mathematics and Its Applications
September 2026 Volume 20 Issue 3 Page 2229-2244
P-ISSN: 1978-7227 E-ISSN: 2615-3017
https://doi.org/10.30598/barekengvol20iss3pp2229-2244

BDAMP AND BAMP OPTIMIZATION OF COMMUNICATION IN SDN-BASED FOG AND CLOUD COMPUTING

Mustafa Hasan Albowarab [1], Nurul Azma Zakaria [2*], Z. Zainal Abidin [3], Fairul Azni Jafar [4]

[1,2]*Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka*
[3] *Fakulti Kecerdasan Buatan dan Keselamatan Siber, Universiti Teknikal Malaysia Melaka*
[4] *Fakulti Teknologi dan Kejuruteraan Industri dan Pembuatan, Universiti Teknikal Malaysia Melaka*
Hang Tuah Jaya, Durian Tunggal, Melaka, 76100, Malaysia

*Corresponding author's e-mail: * azma@utem.edu.my*

Article Info

Article History:

Received: 11th August 2025
Revised: 19th January 2026
Accepted: 9th March 2026
Available online: 8th April 2026

Keywords:

Cloud computing;
Fog computing;
Multi-Objective Optimization;
Multi-Objective Particle Swarm Optimization;
SDN.

ABSTRACT

Software-Defined Networking (SDN) has emerged as a revolutionary paradigm. The integration of SDN within fog networks represents a synergistic convergence of two cutting-edge technologies. With the complexity of SDN serving fog networks, the optimization of communication cost becomes paramount. Addressing the intricate challenges of communication cost optimization necessitates the application of sophisticated methodologies. Multi-Objective Optimization (MOO) algorithms present a robust solution, allowing for the simultaneous optimization of multiple conflicting objectives. By employing MOO, this research proposes a bi-objective optimization model for the intra- and inter-domain communication cost of controller deployment in an SDN-based computing network. The evaluation performed has captured two aspects of the performance of using Binary Angle quantization Multi-objective Particle swarm optimization (BAMP) and Binary crowding Distance Angle quantization Multi-objective Particle swarm optimization (BDAMP) for SDN controllers' deployment. The first aspect is multi-objective-based evaluation, and the second aspect is the SDN network performance. Our developed BAMP and BDAMP have shown superiority over the benchmarks in terms of both aspects. Most importantly, the best performance is achieved by BDAMP in terms of both intra- and inter-communication cost.

How to cite this article:

M. H. Albowarab, N. A. Zakaria, Z. Z. Abidin, and F. A. Jafar, "BDAMP AND BAMP OPTIMIZATION OF COMMUNICATION IN SDN-BASED FOG AND CLOUD COMPUTING", *BAREKENG: J. Math. & App.*, vol. 20, no. 3, pp. 2229-2244, Sep, 2026.

Journal homepage: *https://ojs3.unpatti.ac.id/index.php/barekeng/*
Journal e-mail: *barekeng.math@yahoo.com; barekeng.journal@mail.unpatti.ac.id*

Research Article · Open Access

1. INTRODUCTION

In the ever-evolving landscape of network technologies, Software-Defined Networking (SDN) has emerged as a revolutionary paradigm. By segregating the control plane from the data forwarding plane, SDN offers unparalleled control over network traffic through the use of application programming interface and open interfaces. This separation has not only redefined network management but also fostered the development of innovative applications and technologies, setting the stage for the future Internet [1]. Parallel to the rise of SDN, fog computing networks have surfaced as a vital extension of cloud computing. By decentralizing computation, storage, and control closer to the data sources, fog networks enhance efficiency, reduce latency, and offer real-time processing capabilities. This decentralized architecture is particularly suited for applications requiring immediate data analysis and response, thus filling a critical gap in traditional cloud computing models [2], [3].

The integration of SDN within fog networks represents a synergistic convergence of two cutting-edge technologies. SDN's flexibility and programmable nature align seamlessly with the decentralized structure of fog networks, enabling more efficient network management. By deploying SDN controllers in fog computing nodes, the control of network traffic is optimized, resulting in improved performance, reliability, and scalability. This integration marks a significant advancement in network design, offering new avenues for exploration and development [4]. With the complexity of SDN serving fog networks, the optimization of communication cost becomes paramount [5]. The cost associated with intra-communication (within the same cluster) and inter-communication (between different clusters) can significantly impact the network's overall performance and expenditure. Balancing these costs is a complex task, especially in large-scale networks where the dynamics of network traffic can lead to imbalances in controller load. The challenge lies in determining the optimal number and placement of controllers, tailored to the network topology, to manage network traffic adeptly [6], [7].

Addressing the intricate challenges of communication cost optimization necessitates the application of sophisticated methodologies. Multi-Objective Optimization (MOO) algorithms present a robust solution, allowing for the simultaneous optimization of multiple conflicting objectives [8]. By employing MOO, this research proposes a bi-objective optimization model for the intra- and inter-domain communication cost of controller deployment in an SDN-based computing network. The utilization of MOO not only enhances the efficiency of controller deployment but also contributes to the broader understanding of optimization techniques in the context of SDN and fog networks. In this context, it is important to recognize that SDN deployment is not an arbitrary process. Rather, it constitutes a Multi-Objective Problem (MOP) that requires balancing several objectives concurrently. Most notably, minimizing inter- and intra-communication costs. Consequently, existing MOP-based SDN deployment algorithms have been extensively reviewed, particularly concerning the employed optimization algorithms and the selected objectives.

The study in [9] proposed a dynamic optimization algorithm based on the Salp Swarm Optimization Algorithm for the placement of SDN controllers in large-scale networks. The algorithm evaluates the optimal number of controllers and the optimal connections between switches and controllers. It also incorporates chaotic maps to enhance its performance. The proposed algorithm has been evaluated through several experiments compared to linear and metaheuristic algorithms in terms of execution time and reliability. Meanwhile, the work in [10] proposed a hybrid optimization algorithm, Hybrid Differential Evolution and Whale Optimization, for optimizing the placement of SDN controllers in Internet of Things (IoT) enabled smart city networks. The algorithm searches for optimal locations of controllers while balancing switch loads and minimizing link failures and end-to-end delays. This study suggests that the proposed algorithm can overcome the challenges of controller placement in SDN-IoT networks and improve network performance.

In [5], an architecture for smart city networks that combines cloud data centers, SDN controllers, fog controllers, fog devices, and smart sensors was proposed. To address the challenge of managing large amounts of data from IoT devices, the authors propose the use of fog computing, where computational and storage capabilities of end devices are utilized. An integer programming model is formulated for deploying fog nodes, fog controllers, and SDN controllers while minimizing latency, traffic, and cost. The problem is solved using the weighted sum method and two metaheuristic algorithms, the genetic algorithm (GA) and particle swarm optimization algorithm (PSO). Research in [11] addressed the challenge of placing SDN controllers in networks under conflicting network performance metrics. The proposed solution involves an MOP that considers both latency and bandwidth in the objective function. Two metaheuristic optimization

algorithms, the Non-dominated Sorting Genetic Algorithm II (NSGA-II) and the Pareto Simulated Annealing, are compared in terms of accuracy, processing time, and computational resource utilization.

In addition, the approach in [12] resolved the Multi-controller SDN optimal placement problem by using a hybrid metaheuristic algorithm. The approach starts with the construction of an SDN network using graph theory to improve connectivity and flexibility between switches and controllers. Then, the firefly optimization algorithm is used to select an optimal controller from multiple controllers based on controller features. Finally, the hybrid metaheuristic algorithm consisting of a harmonic search algorithm and PSO algorithm is employed to perform multi-controller placement considering location and distance. The work presented in [13] proposed a meta-heuristic strategy called Capacited Controller Placement Grey Wolf Optimization to find optimal positions for multiple controllers in a software-defined network. The aim is to prevent a drastic increase in worst-case latency in the event of link failure. The proposed approach utilizes the Grey Wolf Optimization algorithm, which is capable of handling MOP. The authors highlight that the worst-case latency is greatly influenced by the number and placement of controllers in the network, and planning for failure is necessary to ensure network reliability.

Furthermore, the authors in [14] developed a PSO algorithm, which is used to optimize the switch-controller average transmission delay for effective deployment of multiple controllers in a continuous two-dimensional space in Software Defined Networks to improve network reliability and resilience. The algorithm abstracts the multi-controller deployment scheme into particles to improve the global search capability and convergence accuracy of the PSO algorithm. Study in [15] proposed a meta-heuristic algorithm with new iterations and tempered mating conditions to solve the Controller Placement Problem, the Garter Snake Optimization Capacitated Controller Placement Problem, for the controller placement problem in Wide Area Networks and SDN. The objective is to reduce end-to-end latencies by carefully placing controllers while considering the capacity of the controllers.

Moreover, the work in [16]–[20] described a dynamic multi-controller deployment scheme based on load balancing. The flow requests in [16] are modeled as a queuing system considering traffic delay and controller capacity, and a PSO-based modified affinity propagation algorithm is proposed to improve clustering performance and achieve efficient network planning. In [17], a hierarchical multi-controller strategy for SDN-based 6G SAGIN is proposed, where simulated annealing optimizes controller placement, and a switch migration mechanism dynamically redistributes loads to maintain performance under changing satellite topologies. Similarly, [18] introduces MC-LBTO for SD-IoT, which employs adaptive coordination among distributed controllers and programmable data plane intelligence to achieve latency-aware load balancing, significantly reducing request delays and improving throughput. In contrast, [19] focuses on algorithmic optimization through Genetic-Bird Swarm Optimization (GBSO), which dynamically selects switches and controllers to balance workloads, outperforming traditional methods in accuracy and efficiency. Meanwhile, [20] addresses multi-controller deployment in Space-Ground Integrated Networks (SGIN) using an improved immune-based heuristic algorithm that accelerates convergence while considering control delay and resource constraints for balanced load distribution.

While existing research has explored dynamic multi-controller deployment and load balancing in SDN-based architectures, most approaches optimize single objectives or rely on heuristic methods, leaving a gap in addressing multiple conflicting objectives simultaneously. This study bridges that gap by introducing an MOO framework that jointly minimizes intra- and inter-domain communication costs for SDN controller deployment in fog computing environments. Specifically, we propose two novel algorithms, BAMP and BDAMP, to enhance solution diversity and convergence. Unlike prior works, this approach integrates angle-based quantization and crowding distance mechanisms within a binary PSO framework, enabling superior optimization of controller placement under complex network conditions. Experimental results confirm that BDAMP achieves the best performance in reducing both intra- and inter-domain communication costs, outperforming benchmark algorithms in multi-objective evaluation and overall SDN network performance. This combination of MOO with advanced PSO variants has not been previously applied in SDN fog computing contexts, marking a significant contribution toward efficient, scalable, and resilient next-generation network architectures.

The rest of the article is structured as outlined below. Section 2 introduces the methodology. Following that, in Section 3, we detail the results and discussion of the study. Finally, Section 4 includes the conclusion and future directions.

2. RESEARCH METHODS

This section presents the proposed solution for enabling the deployment of SDN controllers for the computing network. The deployment is based on the joint optimization of the intra and inter communication cost, known as IIC. The methodology is decomposed into several sub-sections. First, we present the system architecture in sub-section 2.1. Next, the problem formulation is presented in sub-section 2.2. The mathematical optimization is described in sub-section 2.3. Next, the algorithm is outlined in sub-section 2.4.

2.1 System Architecture

The system consists of a computing network interconnected with a set of switches and SDN-controllers. As it is shown in Fig. 1, the computing nodes are interconnected to enable collaboration among them for executing various computing tasks. Hence, a routing mechanism between the nodes is needed. The routing is done by enabling the OpenFlow protocol, in which the flow table is updated by the controller at each switch. In order to assure optimum performance, a deployment process has to be activated. The role of the deployment is to define for each switch its controller that handles the requests from the switch, guaranteeing minimum intra and inter communication cost.

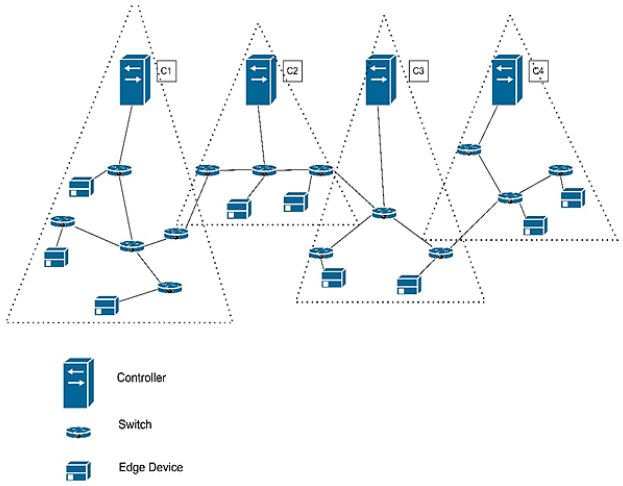

Figure 1. Conceptual Diagram of the Computing Network Supported by SDN Communication

2.2 Problem Formulation

Charges for both intra-domain and inter-domain communication are included in multi-controller implementation costs. The cost of intra-domain communication increases as more controllers are employed, whereas the cost of inter-domain communication decreases. To preserve network performance, it is vital to ensure that the propagation delay between switches and controllers does not exceed a specific threshold. The processing power and bandwidth limitations of the controller also have an impact on the number of switching requests that it can handle. When allocating switches, the controller's processing power should be considered. Due to controller capacity and traffic propagation delay, the formulation of multi-controller deployment in SDN is as follows: Assume that the network's switch and link architecture is known in order to divide an SDN network into suitable control domains and identify the switches in each domain in a way that minimizes total intra-domain and inter-domain communication costs. A single controller should be able to handle no more switch requests than its processing capacity, and the average latency from switches to controllers should not be more than an acceptable level. The controller deployment should also be as evenly distributed as possible.

More formally, it is assumed that a network composed of a set of switches is connected to multiple SDN controllers. An undirected graph $G = (V, E)$ is used to depict the controller network topology, where V and E are the sets of nodes and links, respectively. M represents the number of controllers in the network and C is the set of controllers, where $C = \{C_1, \dots, C_M\}$. N represents the number of switches and S is the set of switches, where $S = \{S_1, \dots, S_N\}$. Therefore, $|V| = M + N$.

Each controller has a capacity, A, which is determined by factors such as CPU, bandwidth, and memory, $A = \{A_1, \dots, A_M\}$. L_M is the redundancy factor of each controller which ranges from 0 to 1, indicating that the controller has sufficient capacity to perform state synchronization. Based on the

characteristics of OpenFlow switch sending requests to the controller, it can be reasonably assumed that the arrival times of new flow set-up requests constitute a Poisson process with rate Φ_j^t, and the processing times of flow requests are independent and identically distributed with negative exponential distribution, with a mean of $\frac{1}{A_j}$.

The subordinate controller receives packet-in information from the switch and calculates the forwarding path and flow table entries. Using principles of queuing theory, the transmission and processing of traffic are described by an $M/M/1$ queue [21]. Flow requests are then aggregated on the connected switch in the form of a queue. Eq. (1) illustrates the calculation for the average sojourn time of the j^{th} controller.

$$\omega_j^t = \frac{1}{A_j - \Phi_j^t}. \tag{1}$$

Given that the time of computing a single source route is subject to the network size, the average response time of the j^{th} controller can be calculated as shown in Eq. (2).

$$\Delta t_j = |V|^2 \cdot \omega_j^t \tag{2}$$

Where $|V|$ represents the number of nodes in SDN, which indicates that the processing time is affected by the network size. Therefore, the average controller response time in the time slot t between the switch and controller can be calculated using Eq. (3).

$$D^t = \frac{\sum_{j=1}^{M} \Phi_j^t . \Delta t_j}{\sum_{j=1}^{M} \Phi_j^t}. \tag{3}$$

The goal is to determine the best values of x_{ij}^t, where $x_{ij}^t = 1$ indicates that the i^{th} switch is connected to the i^{th} controller in time slot t, to minimize two communication costs. The two communication costs between the switch and controller in the OpenFlow network can be expressed as the communication cost between the switch and controller D_{req}, as shown in Eq. (4).

$$D_{req} = 2v_r \sum_{j \in M} \sum_{i \in N} \left(\left\lceil \frac{\lambda_i^t}{v} \right\rceil d_{ij} x_{ij}^t \right), \tag{4}$$

where v_r is the average rate of polling switches, which is related to the average number of links connected to switches, v is the unit rate, v =1Kb/s, λ_i^t is the request rate of the i^{th} switch in slot time t, and d_{ij} represents the shortest distance between the i^{th} switch and j^{th} controller.

The communication cost caused by the interactive state information between the controllers D_{syn} is given by Eq. (5).

$$D_{syn} = v_s \sum_{j \in M} \sum_{k \in M} d_{jk}, \tag{5}$$

where vs denotes the average transmission rate of the state of the controller.

2.3 Mathematical Optimization

As presented in the previous section, the ultimate goal is to divide the network into reasonable control domains and identify each domain's switches in a manner that minimizes both the intra-domain and inter-domain communication costs, as provided by a set of mathematical equations from Eqs. (6) to (10).

$$x_{ij}^t = argmin\left(D_{syn}, D_{req}\right), \tag{6}$$

$$\sum_{j \in M} x_{ij}^t = 1, \quad \forall i \in N, \tag{7}$$

$$\Phi(t)_j \leq L_j A_j, \quad \forall j \in M, \tag{8}$$

$$D(t) \leq \delta, \tag{9}$$

$$x_{ij}^t \in \{0,1\}, \quad \forall i \in N, \forall j \in M. \tag{10}$$

The constraint in Eq. (7) ensures that each switch can only choose one controller as its master controller. The constraint in Eq. (8) ensures that the load of each controller is within the normal range and does not exceed its processing capacity. The constraint in Eq. (9) shows that the average delay (response time) from each switch to its controller is less than δ. The constraint in Eq. (10) ensures that each switch is connected to a main controller.

2.4 Algorithm

The solution space and objective space are needed in order to carry out the optimisation of the issue discussed in the previous section. A vector $\boldsymbol{x}$ $(1 \times MN)$, where M stands for the number of controllers and N for the number of switches, defines the solution space. For conducting the optimization, two variants that were presented for binary particle swarm optimization are used, namely, BDAMP and BAMP. It is observed that the initial B is used to describe the algorithm; this is because of the binary nature of the solution space.

2.4.1 Network Topology Reading

The pseudocode for creating a network topology is outlined in **Algorithm 1**. The algorithm takes in an argument representing the original topology that comprises a set of switches to be used in the topology. The algorithm creates a capacity vector that indicates the capacity of all controllers $A = \{A_1, \dots, A_M\}$. It also generates the request rate of each switch λ_i^t. The algorithm generates the redundancy factor L_M based on a random number between 0.9 and 1. The algorithm reads data from an input $txtfile$ that provides information such as the number of controllers, the number of switches, and the connections between switches. The algorithm creates a topology-representing struct and fills it with attributes like the number of controllers M and switches N, capacity A, redundancy factor, and $lambda\ \lambda_i^t$.

Algorithm 1. Read Topology Algorithm

Input:
$txtfile$ // text file to read the topology of switches.
Output:
$topology$

Start
2 capacity = capacity vector for all controllers
3 lambda = generate a random number for all switches
4 redundancy Factor = generate a random number between 0.9 and 1.0 for all controllers
5 read from $txtfile$ that contains the number of controllers, the number of switches, and the connection between switches
6 topology = struct ()
7 topology. number of controllers = number of controllers
8 topology. number of switches = number of switches
9 topology.capacity = ones (topology. number of controllers)*capacity
10 topology. redundancy factor = redundancy factor
11 topology. lambda = lambda
End

2.4.2 Solution Generation

The pseudocode for generating a solution is outlined in **Algorithm 2**. The algorithm receives the topology as input and outputs the solution. The algorithm first generates the value of $L_j A_j$ from the capacity and redundancy factor for each controller. It then checks two conditions for each controller, namely, whether the average response time is less than delta and whether the flow request is less than $L_j A_j$. If all conditions are met, a random switch from the set of switches is chosen to link with the controller; this set of switches is known as the leader switches, as seen in line 5. The algorithm then calculates the shortest path between the switch and the leader in order to connect each switch with the nearest controller. All the generated information is stored in a struct named “solution” and returned.

Algorithm 2. Generate Solution Algorithm

Input:
$topology$
Output:
$solution$

Start
L_jA_j = topology. capacity*1024 * topology. redundancy factor
if average controller response time < delta and flow request < L_jA_j
Then
    leaders = generate random leaders of switches to connect them with controllers
    calculate the shortest path between each switch and the leader
    connections = array for the connection between switches and the controllers, connect each switch to the nearest controllers
    solution = struct ()
    solution. X = connections
    solution. Leaders = leaders
End if
End

2.4.3 Solution Validation

The solution is generated randomly by the optimization algorithm. Hence, it is important to validate it and perform maintenance if needed. The solution validation consists of two sub-sections, namely, the leader validation and the sub-domain validation.

1. Leader Validation

The goal of leader validation and maintenance is to ensure that there are no repeated leaders in the leader vector. The pseudocode for leader validation is outlined in **Algorithm 3**, where it takes in the leaders, which represent a vector of switch leaders, and it returns the same vector after validation and maintenance. The algorithm performs a while loop that iterates through the leaders. It takes the index of repeated leaders and replaces them with other leaders chosen randomly.

Algorithm 3. Leader Validation Algorithm

Input:
leaders: a vector of switch leaders
Output:
Leaders

Start
While
Size (leaders = size(unique(leaders))
Do
    get the index of repeated leaders
    replaces the repeated leaders with other leaders randomly
Endwhile
Return leaders
End

2. Sub-domain Validation

The goal of sub-domain validation is to ensure that each switch belongs to the correct sub-domain. The pseudocode for this process is presented in **Algorithm 4**. The algorithm takes a solution and returns the solution after modifying the connecting array. It consists of three steps: first, it calculates the shortest path between each switch and the switch leaders; next, it connects each switch with the nearest switch leader; and finally, it returns the solution with the modified connection array.

Algorithm 4. Sub-Domain Validation Algorithm

Input:
Solution
Output:
Solution after changing the connection array

Start
1 calculate the shortest path between each switch and switch leaders
2 connect each switch with the nearest switch leader
3 return the solution after changing its connection array
End

2.4.4 Objective Function

Algorithm 5 displays the objective function's pseudocode. The algorithm's inputs are the $solution$, $topology$, unit rate v, average polling switch rate vr, and the average transmission rate of the controller state vs. The $cost$, which stands for the two objective values, is the algorithm's output.

The algorithm starts by calculating the distance between each switch and its controller through the switch leaders. These distances are stored in D_{sc}. Next, the algorithm calculates the distance between each controller and the other controllers. In other words, the distances between each switch leader and other switch leaders are calculated and stored in D_{cc}. The intra-domain communication cost, D_{req}, is then calculated. Similarly, the inter-domain communication cost, D_{syn}, is calculated. Both costs, D_{req} and D_{syn}, are stored in $Cost_1$ and $Cost_2$ respectively.

Algorithm 5. Sub-Domain Validation Algorithm

Input:
$solution$ //individual solution
$topology$
v
vr
vs
Output:
$Cost$ //the two objective values

Start
1 D_{sc} = calculate the distance between each switch and the controller connected
2 D_{cc} = calculate the distance between each controller and the other controllers (the distance between each switch leader and the other switch leaders)
3 D_{req} = intra domain communication cost ($topology.lambda$, v, vr, D_{sc}, $solution.connection$)
4 D_{syn} = inter domain communication cost (D_{cc}, vs)
5 $Cost_1 = D_{req}$
6 $Cost_2 = D_{syn}$
End

3. RESULTS AND DISCUSSION

This section provides a comprehensive overview of the results and the corresponding discussion derived from this study. It includes the experimental design outlined in sub-section 3.1, parameter settings described in sub-section 3.2, evaluation results of SDN controller deployment using MOO metrics in sub-section 3.3, and application-level evaluation results in sub-section 3.4. Finally, sub-section 3.5 discusses the key findings.

3.1 Experimental Design

In this study, BDAMP and BAMP algorithms are evaluated by using Binary Multi-objective Particle swarm optimization (BMP) [22], Binary crowding Distance Multi-objective Particle swarm optimization (BDMP) [23], and Binary NSGA2 (BN2) [24] in terms of SDN controller deployment. MATLAB 2019b is

used to conduct the experiments, and the results are determined by using multi-objective evaluation and communication cost measures. The OS3E topology model [25], which includes 34 nodes, 40 links, and 4 controllers, was used to make the experiment more realistic. The OpenDaylight [26] controller was used, and the required application modules were created on the application layer. The values of the topology-related parameters are listed in Table 1, and Fig. 2 displays the graph that goes along with them.

Table 1. Parameters Used for Creating Topology

Parameter	Value
Links	40
Switch	34
Controller	4
Lower bounds leader switch	4
Upper bounds leader switch	34

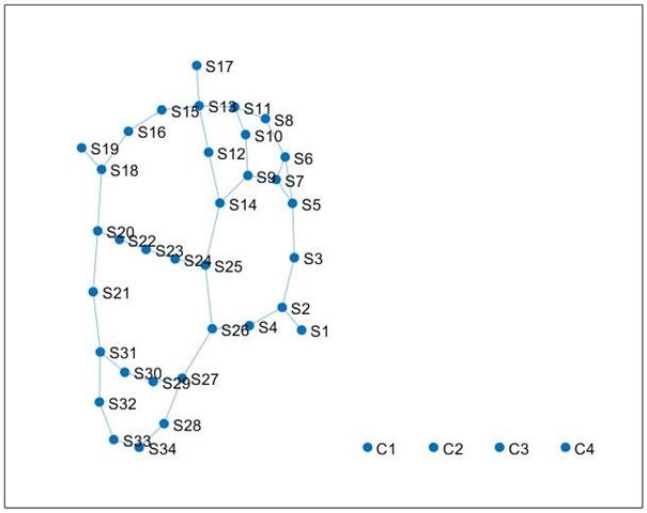

Figure 2. Graph Representation of Network Topology (S denotes Switch, C denotes Controller)

3.2 Setting of Parameters

In the controller deployment process, this parameter is utilised to modify the relative importance of intra and inter communication expenses. The specifics of the SDN controller deployment are shown in Table 2. The parameters used for the deployment of the SDN controller in BAMP, BDAMP, BMP [18], and BDMP [23] are listed in Table 3. Table 4 provides a list of the Binary NSGA2 (BN2) algorithm's input parameters [24]. The same common parameters were used for both the proposed methods and the benchmarks.

Table 2. Parameters of the SDN Controller Network

Parameter	Description	Value
v	Unit rate	1 kb/s
vr	Average rate of polling a switch	10 kb/s
vs	Transmission rate of the state synchronization information of the controller	1 kb/s
LM	Redundancy factor of the controller	[0.9, 1]
$delta$ Δ	The parameter that the average controller response time must not exceed	1 ms
λi lambda	The request rate of i switch	[150, 550] kb/s
N	Number of switches	34
M	Number of Controllers	4
V	Number of controllers and switches	38
A_M	Capacity of each controller	15 M
γ	Weight of the factor	0.8

Table 3. Parameters of BMP, BDMP, BAMP and BDAMP

Parameters	Value
Number of objectives	2
Number of solutions	50
Number of iterations	100
Personal learning coefficient	1/3
Global learning coefficient	2/3
Repository size	100
Inertia weight	0.5
Inertia weight damping rate	0.99
Number of grids per dimension	7
Angular sector width (sigma) for AMP and DAMP only	15
Maximum relative velocity	0.1
Minimum relative velocity	0.001
Each experiment repeated	10

Table 4. Parameters of Binary Non-Dominated Sorting Genetic Algorithm 2

Parameters	Binary NSGA2 (BN2)
Number of objectives	2
Number of solutions	50
Number of generations	100
Crossover option:	
Fraction	2/n
Ratio	1,2
Mutation option:	
Fraction	2/n
Scale	0,1
Shrink	0.5
Each experiment repeated	10 runs

3.3 Evaluation Result of SDN Controller Deployment using MOO Measures

This section presents the multi-objective evaluation measures generated from the experimental design of SDN controller deployment when applied to the developed algorithms and the benchmarks. Furthermore, all of the results are the average of 10 independent runs. First, the Pareto front visualization is presented. Afterwards, the set coverage is compared with the domination percentage among the algorithms.

3.3.1 Pareto Front

The Pareto front is a graphical representation of the set of non-dominated solutions found by the optimization algorithm. This visualization is important for assessing the algorithm's convergence power. In Fig. 3, when the space is divided into two equal partitions: 1 (D_{syn}) and 2 (D_{req}) in partition 1, only four solutions were found by BN2, which dominate over BDAMP and BAMP. However, in partition 2, many solutions found by both BAMP and BDAMP dominate over BN2. This indicates the overall superiority of BAMP and BDAMP. Additionally, BAMP and BDAMP appear to be more powerful with respect to D_{req}, while BN2 appears to be more powerful with respect to D_{syn}. However, BN2 does not generate solutions that are evenly distributed throughout the entire space, in comparison to BDAMP and BAMP.

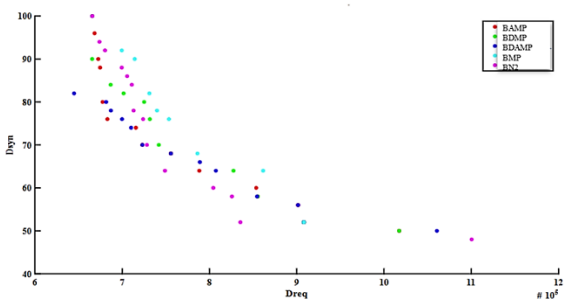

Figure 3. Pareto Front of the Two Optimization Approaches BN2, BMP, BDMP, BAMP, and BDAMP

3.3.2 Set Coverage

The set coverage is generated to demonstrate the relative dominance between the algorithms. As shown in Fig. 4, BDAMP has the highest dominance at 0.91 over BMP, compared to BMP's dominance of 0.007 over BDAMP. Additionally, BDAMP dominance over BDMP is 0.28, which is higher than BDMP dominance over BDAMP at 0.27. Similarly, BDAMP dominance over BN2 is 0.38, which is greater than BN2 dominance over BDAMP at 0.34. Furthermore, BDAMP dominance over BAMP is 0.38, whereas BAMP dominance over BDAMP is only 0.30. This indicates that BDAMP is the most superior algorithm, with BAMP being the second most superior.

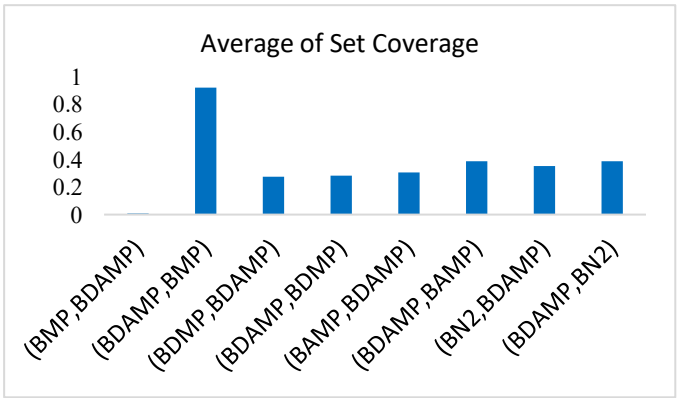

Figure 4. Average Set Coverage between the Approaches of BDAMP Compared with Benchmarks

On the other hand, when comparing BAMP to other algorithms in terms of dominance, Fig. 5 shows that BAMP has a relatively high dominance over BMP at 0.92, compared to BMP's dominance over BAMP at 0.001. Additionally, BAMP dominates BDMP at 0.41, which is higher than BDMP's dominance over BAMP at 0.39. This indicates BAMP's superior performance over other variants of BMP. However, BN2 has higher dominance over BAMP at 0.49, compared to BAMP's dominance over BN2 at 0.37. This is due to the use of crowding distance as an exploration criterion in BN2, which is ignored by BAMP.

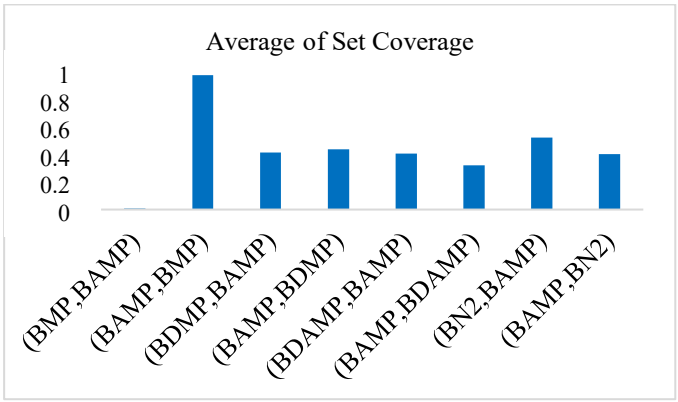

Figure 5. Average set Coverage between the Approaches of BAMP Compared with Benchmarks

3.4 Evaluation Result of SDN Controller Deployment using Application Measures

This section presents the evaluation of the SDN controller deployment measures generated from the experimental design when applied to the developed algorithms and benchmarks. The assessment comprises the verification of the network planning impact of several algorithms under identical circumstances. Additionally, all results are the average of 10 independent runs. The SDN controller load balancing consists of two metrics, namely, intra-communication cost and inter-communication cost. The algorithms produce a set of non-dominated solutions rather than a single optimal one since they are multi-objective. Therefore, one solution from each algorithm is selected, and the results are presented for each metric.

3.4.1 Network Planning

In this experiment, the performance of various network planning algorithms is compared under similar circumstances in order to assess their efficacy. Special attention is given to the subdomain planning because

it affects controller load balancing significantly. In an SDN, a balanced distribution of switches among controllers leads to more efficient subdomain planning and better performance in terms of controller load balancing. To this end, network topologies are generated after controller deployment for each algorithm, providing a clear visual representation of the network division into four domains.

The experimental data provides a summary of the number of switches managed by each controller for the five algorithms in Fig. 6 to further elucidate the comparison of the subdomain planning effects across various techniques. It is shown that the BN2 algorithm has the most variance in the number of switches managed by each controller under the identical controller deployment settings.

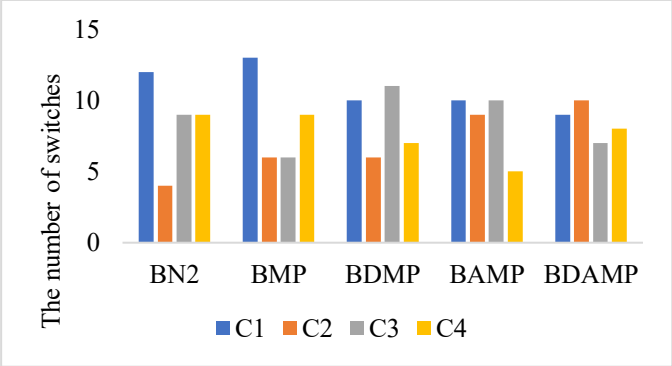

Figure 6. Analysis of the Deployment Results in the Network

In order to calculate a more quantitative measure of balancing, the load on each of the controllers from C1 to C4 is presented with respect to each of the algorithms, as it is depicted in Table 5. Then the difference between the maximum and minimum value for each algorithm is calculated. The minimum value of 3 is obtained by BDAMP, illustrating the best load balancing. On the other hand, the worst solution for load balancing is obtained by BN2, with a value of 8 as the difference between the maximum and minimum number of controllers.

Table 5. Numerical Calculation of the Number of Switches Provided by Each Algorithm and Assigned to Each Controller

Controller	BN2	BMP	BDMP	BAMP	BDAMP
C1	12	13	10	10	9
C2	4	6	6	9	10
C3	9	6	11	10	7
C4	9	9	7	5	8
max-min	8	7	5	5	3

3.4.2 Intra-Domain Communication Cost

The switch sends packets to the controller, which calculates the forwarding path and installs the corresponding flow label on the switch. This process requires communication costs when a new flow table needs to be placed on a switch. On the basis of the flow table, the switch then forwards packets. The switch sends packets to the subordinate controller, and the controller possibly connects with other controllers if the destination switch is in a different sub-domain included in the total flow request path delay in the time domain for the controller. The intra-communication cost D_{req} is the communication cost within one sub-domain between the switch and its subordinate controller, while the inter-communication cost D_{syn} is the cost between two controllers. This cost includes the calculation of the forwarding path and the installation of the flow table on the switch. The results of the intra-communication cost D_{req}, as shown in Fig. 7, indicate that the best performance is achieved by BDAMP with a least observed value of 775577.3, and the worst performance is achieved by BMP with a value of 798352.7.

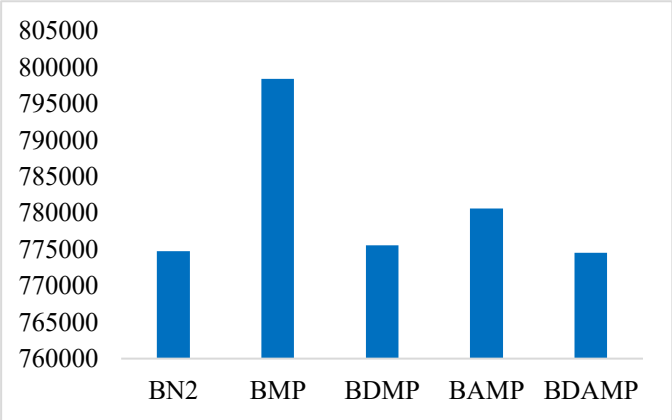

Figure 7. Intra-Domain Communication Cost

3.4.3 Inter-Domain Communication Cost

In a multi-controller environment, it is necessary for controllers to exchange information in order to maintain an accurate view of the overall network status. This exchange of information incurs a cost, known as the state synchronization cost, which primarily consists of the communication costs between controllers. The cost of network connectivity, which includes communication between controllers and switches as well as between switches and controllers, is another factor to consider. The intra-domain communication cost D_{syn} lowers as the number of controllers rises, whereas the inter-domain communication cost rises. In order to minimize both types of communication costs, the network should be divided into appropriate control domains to identify the switches in each domain. An analysis, as shown in Fig. 8, indicates that the least value for intra-domain communication cost D_{syn} is 97.9 for BDAMP and 68.1 for BN2, whereas the highest value is 75.1 for BMP. This suggests that BDAMP performs competitively, whereas BMP performs poorly.

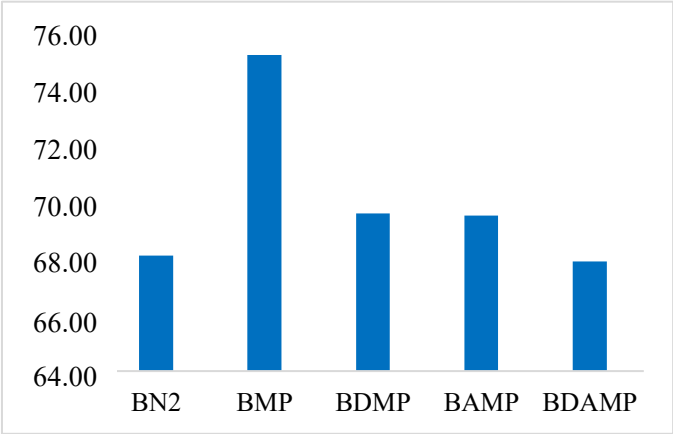

Figure 8. Inter-Domain Communication Cost

3.5 Discussion

The evaluation performed has captured two aspects of the performance of using BAMP and BDAMP for SDN controllers' deployment. The first aspect is multi-objective-based evaluation, and it has shown various observations and findings. First, each of BAMP and BDAMP has shown overall superiority in terms of the Pareto front compared with benchmarks. Second, set coverage evaluation has shown that BDAMP is superior as well. It has accomplished the highest domination, which is 0.91 over BMP, compared with only the domination of 0.007 of BMP over BDAMP. Furthermore, BDAMP has dominated BAMP with a percentage of 0.38 compared with only the domination of BAMP over BDAMP with a value of 0.30. This indicates that the joint usage of crowding distance and angle is more effective than using one of them only. In addition, the usage of angle only is also effective, where BAMP has outperformed other variants of BMP.

In terms of the SDN network performance, the difference between the maximum and minimum values for each algorithm is calculated. The minimum value of 3 is obtained by BDAMP, which has provided the best load balancing. On the other hand, the worst solution for load balancing is obtained by BN2, with a value of 8 as the difference between the maximum and minimum number of controllers. Furthermore, the results

of D_{req} are achieved with the least observed value, which is 775577.3 for BDAMP, whereas the highest value is observed in BMP with a value of 798352.7. Lastly, the least achieved value for D_{syn} is 67.9 for BDAMP, whereas the highest value is observed in BMP with a value of 75.1.

4. CONCLUSION

This article presented a comprehensive methodology for optimizing both intra- and inter-domain communication costs in SDN-based fog computing environments. The controller deployment problem was formulated as a mathematical model and addressed through systematic optimization steps, including topology analysis, solution generation, validation, and objective function design. To address the multi-objective nature of the problem, two novel algorithms, i.e., BAMP and BDAMP, were developed. Experimental evaluations demonstrated that both algorithms significantly outperform benchmark methods in terms of multi-objective optimization quality and overall SDN network performance. Most notably, BDAMP achieved the best results by effectively minimizing both intra- and inter-domain communication costs, while BAMP also showed strong performance in improving solution diversity and convergence. These findings confirm that integrating angle-based quantization and crowding distance mechanisms within a binary PSO framework provides a robust and scalable solution for controller deployment optimization in SDN fog computing networks. Future work will explore additional metaheuristic algorithms and incorporate more realistic network constraints into the optimization objectives to further enhance adaptability and resilience.

Author Contributions

Mustafa Hasan Albowarab: Conceptualization, Methodology, Writing–Original Draft, Formal Analysis, Validation. Nurul Azma Zakaria: Supervision, Conceptualization, Resources, Draft Preparation, Validation. Z. Zainal Abidin: Supervision, Investigation. Fairul Azni Jafar: Writing–Review and Editing. All authors discussed the results and contributed to the final manuscript.

Funding Statement

This work was funded by the Ministry of Higher Education Malaysia and Universiti Teknikal Malaysia Melaka through the Fundamental Research Grant Scheme (FRGS) (FRGS/2018/FTMK-CACT/F00392). We also acknowledge financial contributions from Universiti Teknikal Malaysia Melaka's Zamalah Scheme for sponsoring the scholarship.

Acknowledgment

The authors would like to thank the Ministry of Higher Education Malaysia and Universiti Teknikal Malaysia Melaka for providing financial support for this research through the Fundamental Research Grant Scheme (FRGS) with reference number (FRGS/2018/FTMK-CACT/F00392). The authors would also like to thank the UTeM Zamalah Scheme for sponsoring the scholarship. The authors would like to thank Universiti Teknikal Malaysia Melaka and Fakulti Teknologi Maklumat dan Komunikasi for providing all support for this research.

Declarations

The authors declare that they have no conflicts of interest to report study.

Declaration of Generative AI and AI-assisted Technologies

Generative AI tools (e.g., ChatGPT) were used solely for language refinement, including grammar, spelling, and clarity. The scientific content, analysis, interpretation, and conclusions were developed entirely by the authors. All final text was reviewed and approved by the authors.

REFERENCES

[1] R. Amin, M. Reisslein, and N. Shah, "HYBRID SDN NETWORKS: A SURVEY OF EXISTING APPROACHES," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018. doi: https://doi.org/10.1109/COMST.2018.2837161

[2] H. F. Atlam, R. J. Walters, and G. B. Wills, "FOG COMPUTING AND THE INTERNET OF THINGS: A REVIEW," *big data Cogn. Comput.*, vol. 2, no. 2, p. 10, 2018. doi: https://doi.org/10.3390/bdcc2020010

[3] L.-A. Phan, D.-T. Nguyen, M. Lee, D.-H. Park, and T. Kim, "DYNAMIC FOG-TO-FOG OFFLOADING IN SDN-BASED FOG COMPUTING SYSTEMS," *Futur. Gener. Comput. Syst.*, vol. 117, pp. 486–497, 2021. doi: https://doi.org/10.1016/j.future.2020.12.021

[4] B. P. R. Killi and S. V. Rao, "CONTROLLER PLACEMENT IN SOFTWARE DEFINED NETWORKS: A COMPREHENSIVE SURVEY," *Comput. Networks*, vol. 163, p. 106883, 2019. doi: https://doi.org/10.1016/j.comnet.2019.106883

[5] P. Maiti, H. K. Apat, A. Kumar, B. Sahoo, and A. K. Turuk, "DEPLOYMENT OF MULTI-TIER FOG COMPUTING SYSTEM FOR IOT SERVICES IN SMART CITY," in *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2019, pp. 1–6. doi: https://doi.org/10.1109/ANTS47819.2019.9117921

[6] A. A. Diro, H. T. Reda, and N. Chilamkurti, "DIFFERENTIAL FLOW SPACE ALLOCATION SCHEME IN SDN BASED FOG COMPUTING FOR IOT APPLICATIONS," *J. Ambient Intell. Humaniz. Comput.*, pp. 1–11, 2018. doi: https://doi.org/10.1007/s12652-017-0677-z

[7] M. Hamdan, *et al.*, "A COMPREHENSIVE SURVEY OF LOAD BALANCING TECHNIQUES IN SOFTWARE-DEFINED NETWORK," *J. Netw. Comput. Appl.*, vol. 174, p. 102856, 2021. doi: https://doi.org/10.1016/j.jnca.2020.102856

[8] M. H. Albowarab, N. A. Zakaria, and Z. Z. Abidin, "DIRECTIONALLY-ENHANCED BINARY MULTI-OBJECTIVE PARTICLE SWARM OPTIMISATION FOR LOAD BALANCING IN SOFTWARE DEFINED NETWORKS," *Sensors*, vol. 21, no. 10, p. 3356, 2021. doi: https://doi.org/10.3390/s21103356

[9] A. A. Ateya. *et al.*, "CHAOTIC SALP SWARM ALGORITHM FOR SDN MULTI-CONTROLLER NETWORKS," *Eng. Sci. Technol. an Int. J.*, vol. 22, no. 4, pp. 1001–1012, 2019. doi: https://doi.org/10.1016/j.jestch.2018.12.015

[10] S. K. Keshari, V. Kansal, and S. Kumar, "AN INTELLIGENT WAY FOR OPTIMAL CONTROLLER PLACEMENTS IN SOFTWARE-DEFINED--IOT NETWORKS FOR SMART CITIES," *Comput. Ind. Eng.*, vol. 162, p. 107667, 2021. doi: https://doi.org/10.1016/j.cie.2021.107667

[11] A. C. O. Christofaro, M. M. Carvalho, and D. G. Silva, "PERFORMANCE OF METAHEURISTIC ALGORITHMS FOR THE CONTROLLER PLACEMENT PROBLEM IN SDN," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020, pp. 1–6. doi: https://doi.org/10.1109/CAMAD50429.2020.9209258

[12] N. S. Radam, S. T. F. Al-Janabi, and K. S. Jasim, "MULTI-CONTROLLERS PLACEMENT OPTIMIZATION IN SDN BY THE HYBRID HSA-PSO ALGORITHM," *Computers*, vol. 11, no. 7, p. 111, 2022. doi: https://doi.org/10.3390/computers11070111

[13] K. Kanodia, S. Mohanty, K. Kurroliya, and B. Sahoo, "CCPGWO: A META-HEURISTIC STRATEGY FOR LINK FAILURE AWARE PLACEMENT OF CONTROLLER IN SDN," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020, pp. 859–863. doi: https://doi.org/10.1109/ICICT48043.2020.9112423

[14] Y. Li, W. Sun, and S. Guan, "A MULTI-CONTROLLER DEPLOYMENT METHOD BASED ON PSO ALGORITHM IN SDN ENVIRONMENT," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2020, pp. 351–355. doi: https://doi.org/10.1109/ITNEC48623.2020.9084702

[15] S. Torkamani-Azar and M. Jahanshahi, "A NEW GSO BASED METHOD FOR SDN CONTROLLER PLACEMENT," *Comput. Commun.*, vol. 163, pp. 91–108, 2020. doi: https://doi.org/10.1016/j.comcom.2020.09.004

[16] G. Li, X. Wang, and Z. Zhang, "SDN-BASED LOAD BALANCING SCHEME FOR MULTI-CONTROLLER DEPLOYMENT," *IEEE Access*, vol. 7, pp. 39612–39622, 2019. doi: https://doi.org/10.1109/ACCESS.2019.2906683

[17] Z. Liao, C. Chen, Y. Ju, C. He, J. Jiang, and Q. Pei, "MULTI-CONTROLLER DEPLOYMENT IN SDN-ENABLED 6G SPACE–AIR–GROUND INTEGRATED NETWORK". Remote Sensing, 14(5), 1076, 2022. doi: https://doi.org/10.3390/rs14051076

[18] A. Alyanbaawi, A. El-Sayed, and N. Salah, "MC-LBTO: SECURE AND RESILIENT STATE-AWARE MULTI-CONTROLLER FRAMEWORK WITH ADAPTIVE LOAD BALANCING FOR SD-IOT PERFORMANCE OPTIMIZATION", Scientific Reports, vol. 15, 44660, 2025. doi: https://doi.org/10.1038/s41598-025-31216-6

[19] F. Guo and A. Ye, "THE APPLICATION AND PERFORMANCE OPTIMIZATION OF MULTI-CONTROLLER-BASED LOAD BALANCING ALGORITHM IN COMPUTER NETWORKS", *Egyptian Informatics Journal*, Volume 30, 2025, 100678, ISSN 1110-8665. doi: https://doi.org/10.1016/j.eij.2025.100678

[20] C. Chi, L. Yang, Q. Huang, and Y. Qi, "OPTIMAL PLACEMENT OF MULTI-CONTROLLER CONSIDERING LOAD BALANCE AND CONTROL DELAY IN SOFTWARE DEFINED SATELLITE NETWORK," *2022 34th Chinese Control and Decision Conference (CCDC)*, Hefei, China, pp. 2123-2128, 2022. doi: https://doi.org/10.1109/CCDC55256.2022.10033633

[21] K. Atefi, S. Yahya, A. Rezaei, and A. Erfanian, "TRAFFIC BEHAVIOR OF LOCAL AREA NETWORK BASED ON M/M/1 QUEUING MODEL USING POISSON AND EXPONENTIAL DISTRIBUTION," in *2016 IEEE Region 10 Symposium (TENSYMP)*, 2016, pp. 19–23. doi: https://doi.org/10.1109/TENCONSpring.2016.7519371

[22] C. A. C. Coello and M. S. Lechuga, "MOPSO : A PROPOSAL FOR MULTIPLE OBJECTIVE PARTICLE SWARM OPTIMIZATION," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, IEEE, 2002, pp. 1051–1056. doi: https://doi.org/10.1109/CEC.2002.1004388

[23] F. Sheikholeslami and N. J. Navimipour, "SERVICE ALLOCATION IN THE CLOUD ENVIRONMENTS USING MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION ALGORITHM BASED ON CROWDING DISTANCE," *Swarm Evol. Comput.*, vol. 35, pp. 53–64, 2017. doi: https://doi.org/10.1016/j.swevo.2017.02.007

[24] K. Deb, A. Member, A. Pratap, S. Agarwal, and T. Meyarivan, "A FAST AND ELITIST MULTIOBJECTIVE GENETIC ALGORITHM: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002. doi: https://doi.org/10.1109/4235.996017

[25] I. O. Science, “SCHOLARSHIP AND SERVICES EXCHANGE,” 2020.
[26] Z. K. Khattak, M. Awais, and A. Iqbal, “PERFORMANCE EVALUATION OF OPENDAYLIGHT SDN CONTROLLER,” in *2014 20th IEEE international conference on parallel and distributed systems (ICPADS)*, 2014, pp. 671–676. doi: https://doi.org/10.1109/PADSW.2014.7097868