Int. J. Nonlinear Anal. Appl. 16 (2025) 9, 157–167
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2024.34424.5141

Enhancing quality of service in SDNs through Pareto-optimized controller placement using NS-MF algorithm

Ahmad Jalili

Department of Computer Engineering, Faculty of Basic Sciences and Engineering, Gonbad Kavous University, Gonbad Kavous, Iran

(Communicated by Seyed Hossein Siadati)

Abstract

Software-defined networks (SDN) have emerged as a new paradigm to overcome rigidity in traditional networks. SDN controllers manage network switches through a centralized control plane. Strategically placing controllers is vital for meeting performance needs. We model the NP-hard controller placement problem (CPP) as multi-objective optimization reconciling switch-controller latency, resilience to failures, inter-controller coordination overhead and load balancing. A customized Non-dominated Sorting Moth Flame algorithm (NS-MF) with novel recombination and perturbation techniques is proposed to effectively approximate the Pareto-optimal set of placements on large problem instances. NS-MF is benchmarked on a diverse corpus of 41 topologies against the exhaustive POCO solver, assessing computational time and solution quality tradeoffs. Compared to POCO, the proposed algorithm attains over 20X speedup for the largest graphs with an average optimality gap within 0.8%. The proposed NS-MF demonstrates superior performance over state-of-the-art metaheuristics (NSGA-II and PSA) in reconciling proximity and diversity objectives when estimating Pareto-optimal fronts. Experimental results substantiate NS-MF's efficacy in effectively navigating objectives pertinent to resilient SDN design.

Keywords: Facility Location Problem (FLP), Multi-Objective Combinatorial Optimization (MOCO), Software Defined Networking (SDN), Controller Placement Problem (CPP), Non-dominated Sorting Moth Flame algorithm (NS-MF), Heuristic Algorithms.
2020 MSC: 90C27, 68T20

1 Introduction

Software Defined Network (SDN) is a new networking approach that separates the control plane from the data plane, offering advantages such as flexibility and scalability [17]. The controllers, which are logically centralized, administer the switches by establishing rules for packet management. However, there is a need for physical distribution of controllers to meet performance, scalability, resilience, and fault tolerance requirements. To address these challenges, concepts like HyperFlow have emerged, allowing for the division of SDN architecture into multiple domains managed by individual controllers [15]. In large-scale WAN deployments, the traditional centralized approach of a single controller poses limitations in terms of performance and scalability. Recent suggestions propose the use of multiple controllers working together to handle the network.

One critical aspect in SDN is the placement of controllers, which involves locating a limited number of controllers in a network to meet various requirements [9]. Interestingly, this problem maps precisely to the classical "facility

Email address: jalili@gonbad.ac.ir (Ahmad Jalili)

Received: June 2024 *Accepted:* August 2024

location problem" (FLP) that has intrigued optimizers for decades. Like determining optimal plant locations or fire station placements, controller positioning requires balancing multiple objectives - latency, load, redundancy - across a graph of nodes [10].

The controller placement problem, recognized as NP-hard, can be framed as a multi-objective combinatorial optimization challenge, where the solution involves a trade-off between multiple optimization parameters. Previous approaches have focused on propagation latency between controllers and switches, but they often overlook inter-controller latency. This factor is crucial in concrete networks, and objectives such as latency between switches and their assigned controllers, inter-controller latency, and failure need to be considered [1]. Therefore, the controller placement problem can be expressed as a multi-objective combinatorial optimization challenge, allowing for a clear demonstration of trade-offs between competing criteria.

Discovering a viable solution for the controller placement problem is essential in designing an efficient and reliable SDN-based WAN architecture. Stochastic The placement of controllers yields unsatisfactory performance, so careful planning is necessary. For small networks, exhaustive evaluation of all potential placements ensures optimal results based on the metrics. These optima can serve as guidelines for sizing the control plane.

Ensuring a controller placement that strikes a balance between use case objectives is crucial for efficient operation. Furthermore, for large networks, an efficient and computationally fast approach is necessary to solve the placement problem. While exhaustive evaluation is possible for small and medium-sized networking, it becomes impractical for large ones. Here, heuristic approaches are explored, specifically the Non-dominated Sorting Moth Flame algorithm (NS-MF), to efficiently solve the multi-objective CPP. While heuristic approaches may not guarantee optimal solutions, they can provide accurate estimations of the Pareto optimal set. The NS-MF algorithm is capable of finding accurate estimations of the Pareto optimal solution, producing results in faster computation times and with significantly less memory requirements [21, 22]. It is particularly effective in addressing conflict objectives and large-scale networking with numerous solutions.

In summary, this work introduces a modified version of the NS-MF algorithm to solve the NP-hard multi-objective CP problem in SDN. Several new mechanisms, including a greedy initialization method, non-dominated sorting, fast Pareto finder procedure, path-relinking strategy, and dispersion mechanism, are added to the adapted NS-MF. Results demonstrate that the introduced algorithm provides a strong approximation of the Pareto optimal set based on the defined objectives. Compared to the POCO framework [5], the proposed algorithm can find near-optimal solutions in a much shorter time and with less memory. Its efficiency is particularly evident in large SDN networks where the POCO framework exceeds the capacity of the system's RAM.

The rest of the paper is organized as follow. Related works of the controller placement problem are presented in following section. In Section 3, the detail of problem is defined. The proposed Heuristic Algorithm and a thorough presentation of its operators are brought in Section 4. Section 5 explores the result and analysis. Conclusion and future works are explained in final section.

2 Related Works

The placement of SDN controllers is an essential consideration in optimizing network performance. Several papers propose different approaches to address this issue. Jeya et al. introduce the use of SDN architecture in smart grids and propose a method to find the optimal solution for controller placement in a smart grid using SDN architecture [12]. The paper discusses the challenges of placing controllers in a smart grid implementing SDN architecture, but it does not specifically mention the use of latency and other metrics for controller placement.

Dou et al. propose a solution for optimizing control latency in software-defined wide area networks (SD-WAN) by considering the ability to program critical flows at switches [3]. In this article, the authors proposed a control latency optimization algorithm based on the ability to program of important flows at switches, we employed this metric to determine the optimal placement of controllers and establish mappings between nodes and managers. Proposed method reduces control latency by up to 62.3%, 27.5%, 58.3%, and 61.7% under different topologies. However, the paper does not present a detailed discussion of the important factors and considerations that are taken into account when evaluating the programmability of critical flows at switches.

Authors in [25] propose a deep reinforcement learning-based model to dynamically adjust the controller placement in a virtualized environment to minimize OpenFlow latency. In this article , a novel deep reinforcement learning model is introduced to dynamically adjust the controller's position, aiming to minimize delay within a virtualized environment. Outcomes reveal that the introduced method outperforms both random and generic strategies. The

primary objective of the model is to enhance performance in a virtualized setting, but it did not explore other latencies metrics.

In [19], a thorough investigation of the MCPP in Software-Defined Networking is conducted, focusing on aspects such as load balancing, reliability, and dynamic techniques. The study encompasses a categorization of these techniques, simplifying researchers' grasp of their underlying principles. The paper contributes to addressing the network management issues faced by traditional networking in the context of increasing Internet usage and evolving technologies.

A novel approach, called reliability-aware swarm-based multi-objective optimization, is introduced for strategically placing controllers in distributed Software-Defined architecture [7]. The goal is to improve the network performance, scalability, and fault tolerance by using a heuristic algorithm and a hybrid PSO algorithm. The paper argues that the existing solutions for CP problem in SDN networks often neglect the reliability, cost-effectiveness, and dynamic resource allocation aspects, which are essential for real network deployment. The paper aims to address these challenges by considering the failure probabilities, latency, load balancing, and controller capacity constraints in the optimization problem. The paper also develops a heuristic CPP algorithm and a hybrid RDMCP-PSO algorithm to solve the RDMCP problem efficiently and effectively. However, the paper does not consider the communication overhead and synchronization issues among multiple controllers, which may affect the network performance and reliability.

Ma and et. al. introduce an improved optimization algorithm, named IAVOA, for solving CPP in distributed SDN. The algorithm, derived from the African Vultures Optimization Algorithm (AVOA), utilizes the multi-node and two-node relocation methods to enhance convergence speed. Comparisons with other meta-heuristic algorithms reveal that IAVOA achieves the lowest average latency between controllers and switches in shortest, improving convergence speed by 38.16% compared to the best-performing algorithm. The research contributes to the field by developing IAVOA, applying it to solve CPP in SDN, and showcasing its superior performance across various real topologies. The limitations of this study include the study only focuses on optimizing the delay between controllers and switches in the CPP. Other objectives, such as controller load and energy consumption, are not considered. The study explores the situation of one switch connected to only one manager. It does not explore scenarios where multiple switches are connected to a controller or where one switch is connected to multiple controllers [16].

The authors in [18] propose a linear programming method to attain a trade-off between setting-up cost and delay in the placement of SDN controllers in large scale networks. The introduced method specifies the location, and the minimum number of demanded controllers based on the topology, processing capacity of controllers, and setup cost. The result shows that proposed method attains the lowest setting-up cost and average delay of controlling traffic compared to previous methods. It also identifies the switches associated with each suggested controller in actual network topologies. However, the study did not consider other factors such as reliability, energy consumption, or load balancing.

3 Problem Definition

The CPP focuses on strategically positioning control units within a network, considering both the network's layout and the required number of controllers. Users can define specific criteria to guide where controllers are placed. Minimizing latency between each device and its assigned controller is a central goal, but other factors also need to be balanced, like: Node or link failures: Ensuring the network remains functional even when parts fail. Controller-to-controller latency: Maintaining efficient communication between control units. Load balancing: Distributing workload evenly across controllers for optimal performance [11].

These factors combine to create a multi-objective CPP. We can represent the network as a graph $G = (V, E)$, with: V: A set of n nodes (devices) connected by links (edges) in set E. Moreover, a distance matrix storing the shortest path based on delay between each node, where $D(x, y)$ determines the latency between nodes x and y. Latencies are normalized using the graph's diameter (largest distance). Before solving the CPP, we need to determine the desired number of controllers (k). This creates a search space of k-element subsets of n nodes, where each subset represents a potential controller placement. For example, with 41 nodes and $k = 7$, the subset {3, 7, 19, 21, 28, 32, 35} signifies placing controllers at nodes 3, 7, 19, 21, 28, 32, and 35.

In a network with n nodes and k desired controllers, rearranging controllers within a chosen set doesn't create new options. This means there are only $C(n, k)$ possible placements, where $C(n, k)$ represents the number of ways to choose k elements from n. This set of all possible placements forms the "search space" for the CPP. Our goal is to find the Pareto optimal within this search spaces. A solution location is optimal if does not exist other solution where all objectives improve simultaneously. In practice, we often minimize multiple objectives (metrics) like latency and

load balancing.

The set of objective values for all Pareto optimal solutions forms the Pareto frontier (PF). Ideally, finding the PF would involve evaluating all $C(n,k)$ options, which is infeasible for large networks due to the massive memory and time demands. The POCO framework exhaustively evaluates every single placement in the search space, leading to high memory and time demands for large networks. This makes it impractical, so researchers often switch to heuristic approaches for faster solutions. However, this comes at the cost of potentially less accurate results. Different measures, drawn from [5], are used to assess the gap between the actual and approximated Pareto frontiers.

3.1 Objective Functions

Choosing the best placement for controllers in a network requires considering several competing factors, like how long data takes to travel between devices and controllers (switch-to-controller delay) and between controllers themselves (inter-controller delay). This paper focuses on two key aspects of each type of latency: Switch-to-Controller Latency: Worst-Case: Defined in Equation (3.1), this metric represents the maximum time data takes to reach any controller from a specific device. It's crucial for network performance and is minimized in most approaches ([6, 25]). Average: Defined in Equation (3.2), this metric calculates the average time data takes to reach any controller from all devices. It's another vital factor impacting scalability, fault tolerance, and overall network interaction. Inter-Controller Latency: Worst-Case: This represents the maximum time data takes to travel between any two controllers, impacting coordination between them. Average: This calculates the average time data takes to travel between all controller pairs. These latencies are significant because they directly affect various aspects of network performance. Existing approaches often rely on propagation latency as the main input for measuring latency during controller placement [6, 25].

$$f_{\text{worst\_}N2C} = \max_{v \in V} \min_{p \in P} d_{v,p} \tag{3.1}$$

$$f_{\text{avg\_}N2C} = \frac{1}{|V|} \sum_{v \in V} \min_{p \in P} d_{v,p} \tag{3.2}$$

As SDN scales to manage vast networks, dividing it into interconnected domains enhances both scalability and security. However, efficient resource utilization requires communication between controllers across domains. Imagine a request from one domain needing resources in another-the controllers need to exchange network parameters and synchronize states. This inter-controller communication directly impacts the overall performance of communication between two hosts [5, 7]. Therefore, minimizing inter-controller latency becomes another crucial goal in controller placement. The closer you place controllers, the less communication cost and faster the synchronization. Intuitively, we aim to minimize the latency between them. Metrics for assessing both the maximum and mean inter-controller delay are established by Equations (3.3) and (3.4), respectively.

$$f_{\text{worst\_}C2C} = \max_{p1,p2 \in P} d_{p1,p2} \tag{3.3}$$

$$f_{\text{avg\_}C2C} = \frac{1}{\binom{|P|}{2}} \sum_{p_1,p_2 \in P} d_{p_1,p_2} \tag{3.4}$$

3.2 Model

Consider $G = (V; E)$ as the graph illustrating the structure of a network, comprising V nodes and E links. The representation of the model is as follows:

$$\text{minimum} f_{avg\_N2C}, f_{avg\_C2C}(5)$$

subject to:

$$\sum_{i \in V} Z_i = K, (5-1)$$

$$\sum_{i \in V} X_{ij} = 1, \forall j \in V \, (5-2)$$

$$Z_i \in \{0,1\} \, (5-3)$$

$$X_{ij} \in \{0,1\} \, (5-4)$$

Constraint (5-1) guarantees the placement of K controllers. Constraint (5-2) mandates that each switch is under the control of precisely one controller. The binary variables are described in (5-3) and (5-4).

3.3 Performance Metric

We use two sets of solutions to assess the performance of our algorithms: Q, The "gold standard" set of solutions, known as the actual Pareto frontier. This represents the best possible trade-offs between competing objectives. P, The set of solutions generated by our heuristic approach, an approximation of the true Pareto frontier. To quantify how close our solutions are to the ideal, we employ the inverted generational distance (IGD) metric ([8]). This metric has been adapted to our specific problem, following the approaches laid out in [2].

$$IGD\left(Q,P\right)=\frac{\sum_{y\in Q}d\left(y,P\right)}{|Q|} \tag{3.5}$$

where $d(y,P)$ is the minimum Euclidean distance between y and the elements in$P.IGD\left(Q,P\right)$is capable to measure both the diversity and convergence of P.

4 Proposed Heuristic Algorithm (Non-dominated Sorting Moth Flame algorithm)

In this part of the study, we develop a novel multi-objective optimization algorithm named Non-dominated Sorting Moth Flame optimization (NS-MF) designed to handle the placement of surveillance nodes within a network topology. The proposed NS-MF extends previous single and multi-objective moth flame optimization metaheuristics (specifically MFO [13] and NS-MFO [23]) by integrating extra facets like local search operators and perturbations to avoid getting stuck in subpar solutions. Like most evolutionary algorithms, NS-MF takes as input parameters such as maximum generations to determine halting, population size (here number of "moths" denoting candidate node configurations), plus problem-specific extras including connectivity graph, maximum stall limit for Pareto front changes, and randomly generated starting placements to seed the algorithm. The input of the algorithms are it, $S_p, G=(V,E)$, $\max_{it}$ and P_0. We additionally maintain an Pareto-optimal archive external to the population that tracks non-dominated solutions discovered so far, initially empty. Auxiliary inputs consist of bounds on flame numbers derived mathematically [23] and an independent iteration counter for non-improving turns. The procedure initializes by creating an initial random moth population with associated flames equivalent to the predefined population count.

Now to sort moths by layers of non-domination and inject diversity via spacing metrics, the proposed NS-MF procedure adapts the elitist sorting and crowding approaches in the popular NSGA-II genetic algorithm. The end product is a diverse Pareto front showcasing the tradeoffs between the often conflicting surveillance optimization objectives for the decision maker. The enhancements in NS-MF promote exploration and expand usefulness over the native MFO for multi-criteria problems. Individual solutions can be compared and categorized into different non-domination levels. The population contains various fronts $F=\{F_1,F_2\ldots,\}$ (obtained by ranking objective vectors of solutions into layers [24]), where the initial front F_i signifies the level of non-domination or the Pareto front of the solutions. The crowding distance metric serves as a potent method for preserving diversity among the derived solutions. This metric gauges the concentration of solutions around a specific solution within the population. It's computed as the mean distance between two neighboring solutions across each objective.

In this algorithm, FlameMaxNum flames are selected in each iteration as follows: Given the external Pareto set ExtArchive and the current population Pop as well as the non-dominated solutions F_i in the current population, $\lfloor\rho\times$FlameMaxNum$\rfloor$ members are randomly picked from each set to create the set of flames. Here, M contains the best solutions since the start of the search and F1 contains the $\lfloor\rho\times$FlameMaxNum$\rfloor$ best solutions in the current iteration. Hence, selecting flames this way balances intensification towards good solutions and diversification over the whole population to avoid over-focusing on a limited subset of solutions. The movements of the moths in each iteration are guided by these flames.

The path-relinking strategy [20] is utilized as a local search method is employed to refresh the positions of moths and produce the offspring population, termed as OffspringPop. For each moth i, a flame fi is selected from the flame set. Path-relinking is subsequently executed between i and fi to secure the optimal solution along the trajectory extending from i to fi. After that, the parent Pop and child OffspringPop populations are combined.

Non-dominated sorting is applied on the combined population to obtain different fronts F1, F2, etc. The initial front F1 is then added to ExtArchive if no member in ExtArchive dominates it. If a solution is added to ExtArchive, solutions in ExtArchive dominated by the new member are removed to maintain ExtArchive as a non-dominated set. If

no solution is appended to ExtArchive, it indicates that the iteration did not yield any enhancement. Consequently, the counter for no-improvement, IterNoImpr, is increased by one. Next, the Constrained Crowded-Comparison Operator [4] is utilized to sort the combined population. After sorting, the best Pop moths are selected for the next iteration. If IterNoImpr reaches the threshold MaxNoImpr, a perturbation is induced in the population to somehow restart with new moth positions to escape local optima. The final output of the algorithm is the non-dominated solutions in ExtArchive representing the best tradeoffs found.

procedure Non-dominated Sorting Moth Flame (t_max, P_size, $G = (V, E)$, max_it, P_0)

M = ∅

F_max = P_size

counter = 0

while counter < t_max do

Generate a random population of moths.

P = generate_random_population(P_size)

Initialize the maximum number of flames with the population size.

F = P_size

Sort the population in different non-domination levels with computed crowded distance.

P = non_dominated_sorting(P)

Initialize the external Pareto set with the first solution of the population.

M = {P[0]}

For each solution in the population, do

If the solution is not dominated by any solution in M, then add it to M.

if not dominates(P[i], M) then

M = M U {P[i]}

end if

end for

If the external Pareto set has not changed, then stop.

if M == M_old then

break

end if

Update the maximum number of flames.

F = min(F, F_max)

Initialize the counter.

counter = 0

For each solution in M, do

Generate a new solution by perturbing the current solution.

P_new = perturb(P[i])

If the new solution is not dominated by any solution in M, then add it to M.

if not dominates(P_new, M) then

M = M U {P_new}

end if

end for

Update the external Pareto set.

M_old = M

# Select FS flames.
F_flames = [randomly select FS members from M, P, P(F_1)]
# Create the trail.
trail = [P[0]]
# For each flame, do
# Add the flame to the trail.
trail.append(F_flames[i])
# Perform path-relinking between the flame and the current solution.
P_new = path_relinking(P[i], F_flames[i])
# If the new solution is not dominated by any solution in M, then add it to M.
if not dominates(P_new, M) then
M = M U {P_new}
end if
end for
# Update the external Pareto set.
M_old = **M**
end while
return **M**
end procedure

Figure 1. The pseudo code of the proposed heuristic algorithm

5 Result and analysis

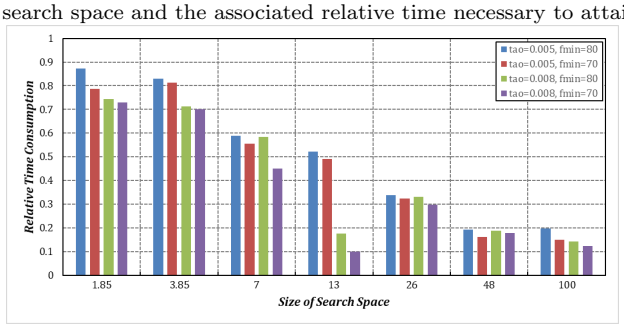
Here, various facets of the algorithm's efficacy are examined. All the tests are executed on a system equipped with an Intel Core i7 processor and 16 GB RAM operating on Windows 10 Pro (64 bit) and Matlab 2020. To analyze the performance of the proposed NS-MF approach, we conduct several benchmarking experiments. Primarily, we aim to quantify the trade-off between computational time savings from our heuristic method and any resulting loss in solution quality. As a baseline, we first exhaustively evaluate a large set of network topologies from the Internet Topology Zoo [14] using the POCO solver to generate reference Pareto fronts. We then run our NS-MF on the same topologies while systematically varying algorithm parameters and topology properties. By comparing NS-MF outcomes against the POCO reference fronts under diverse settings, we elucidate the relationships between accuracy, runtime, and input conditions. These findings let us provide practical guidelines for operators to tune NS-MF to suit their accuracy needs and time constraints when deploying our method on real-world topologies. In summary, our comprehensive benchmarking quantifies the speed vs quality tradeoffs attained in practice by the proposed NS-MF in contrast to exhaustive search.

5.1 A. Execution Time

We conduct timing analyses to gauge the computational performance of the proposed NS-MF algorithm relative to exhaustive POCO evaluation in a machine-independent manner. Figure 2 summarizes benchmark outcomes across seven trial instance categories differentiated by search space size. The horizontal axis denotes bin thresholds in terms of number of possible placements. The initial bin encompasses all scenarios where the search space holds up to 1,850,000 placements, while the second bin includes all scenarios for which $1,850,000 < \binom{n}{k} \leq 3,850,000$ and etc. The vertical axis plots normalized runtime ratios - specifically, the clock time required by NS-MF divided by that for exhaustive POCO enumeration, $t_{\text{relative}} = \frac{t^{NS-MF}}{t^{POCO}}$. Normalizing run times thus allows portable assessment of the relative computational savings afforded by the NS-MF heuristics across problem instances and platforms.

As evidenced in Figure 2, average normalized runtime for NS-MF keeps reducing with increasing instance size when moving right across search space magnitude bins - a direct outcome of amortization of the metaheuristic's overhead.

Figure 1: Dimensions of the search space and the associated relative time necessary to attain different performance tiers.



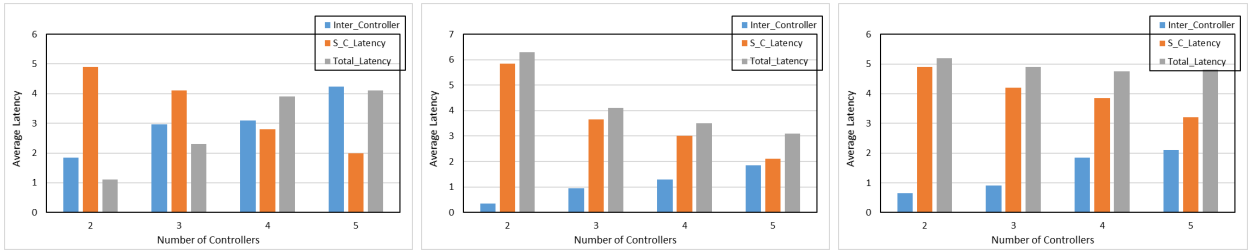


Figure 2: Average Case latency (a) Niif, (b) Arpanet19723 and (c) Grnet topologies.

Specifically, for the two precision targets of 1% and 2%, relative run times for NS-MF drop below 20% for bin categories encapsulating search dimensions up to 14 million and 7.7 million placements respectively.

Thus, as gleaned from the time-accuracy tradeoffs, NS-MF attains over 20X speedup over exhaustive POCO evaluation for the largest instances with average optimality gaps still within 0.8%, thereby providing good approximations of true Pareto fronts. In a nutshell, the efficacy of the proposed NS-MF algorithm at harvesting near Pareto-optimal solutions with negligible optimality loss yet orders of magnitude faster time-to-solution than enumeration is clearly manifested, especially for larger problem scales.

5.2 B. Performance based on Latency

Figures 3 and 4 present experimental results depicting the tradeoffs between average and worst-case latency vs. number of controllers for the Niif, Arpanet19723 and Grnet topologies. The overall end-to-end latency comprises switch-to-controller delay and inter-controller coordination overhead. As expected, increasing controller count reduces average and worst-case switch-to-controller latency due to shorter paths. However, the opposing trend manifests in inter-controller latency as more controllers lead to greater coordination overhead.

For the Arpanet19723 topology specifically, the reductions in switch-controller latency dominate, yielding declining overall latency with more controllers both on average and for worst-case – a trend clearly visible in Figure 2 and the Arpanet19723 column of Table II. However, for Niif and Grnet, coordination overhead increments outweigh reductions in switch-to-controller delay. Consequently, unlike Arpanet19723, total latency for these two topologies remains largely flat or rises marginally as controller count goes up, quantified through in Table I. In summary, our characterization elucidates the complex tradeoffs between localization and coordination impacted by controller deployment density, underscoring topology-specific analysis to balance the two factors.

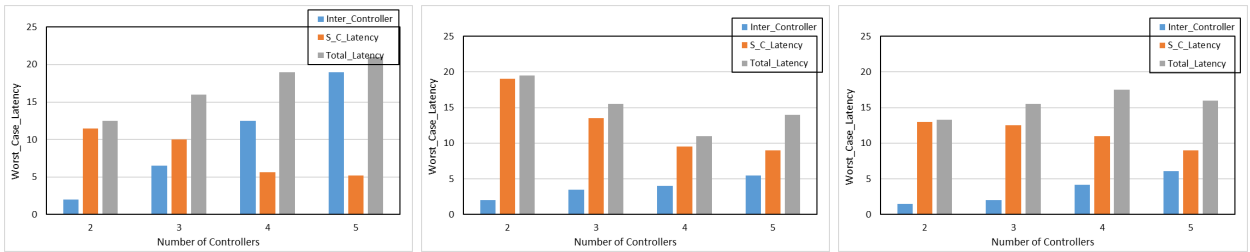


Figure 3: Figure 3. Worst Case latency (a) Niif, (b) Arpanet19723 and (c) Grnet topologies.

Table 1: Table 1. Network structure (overall mean and maximum latency) and duration of execution

Topology	k	Avg_Latency	Exe_Time (ms)	W_C_Latency	Exe_Time (ms)
Niif	2	6.5432	0.0185	12.246	0.1469
	3	6.3214	0.0951	13.965	0.9532
	4	6.2313	0.8621	16.329	1.065
	5	6.095	1.0325	19.032	1.3652
Arpanet	2	6.894	0.3659	17.325	0.954
	3	5.623	1.0314	16.325	1.351
	4	4.0946	6.6241	16.098	4.6851
	5	3.547	13.0325	12.354	9.653
Grnet	2	5.0321	0.9546	13.316	3.651
	3	4.9635	3.0234	13.985	9.684
	4	3.2681	8.6547	14.032	15.663
	5	3.0681	12.3679	14.703	28.681

Table 2: Table 2. Mean and Standrad deviation of IGD

Topology Algorithm	ARN N=28 k=7	Digex N=31 k=6	NetworkUsa N=35 k=7	Chinanet N=38 k=7	Litnet N=39 k=6	Carnet N=41 k=8	Ntelos N=47 k=6	Bellcanada N=48 k=6	Dfn N=51 k=7
NSMF	M: 0.00542	0.02578	0.03098	0.02869	0.01692	0.01529	0.01937	0.27813	0.09246
	SD: 0.00213	0.01091	0.01847	0.01094	0.01034	0.02811	0.01518	0.01834	0.07926
MHNSGA	M: 0.04561	0.34097	0.39027	0.40948	0.56137	0.60873	0.49156	0.72308	0.62901
	SD: 0.03567	0.0658	0.07013	0.07498	0.26166	0.10935	0.10375	0.19035	0.09376
PSA	M: 0.0967	0.40961	0.40981	0.75012	0.89267	0.83461	0.72902	0.67276	0.92803
	SD: 0.0173	0.09237	0.08495	0.08926	0.19257	0.18247	0.17493	0.18726	0.20046

5.3 C. Comparative Study and Analysis of Convergence

As discussed in Section 3.3, the inverted generational distance $IGD\,(R, M)$ metric quantifies Both the diversity and convergence of an estimated Pareto set in comparison to the actual Pareto front. Since multi-objective algorithms are evaluated on test problems with known optimal Pareto sets (R), IGD values measure closeness of convergence, with lower scores being better. We now present further convergence analyses benchmarking NS-MF against existing methods from the literature using IGD. Specifically, 9 network topologies from the Internet Zoo dataset are adopted as test cases, with node counts (N) and controller numbers (k) varied as shown. Algorithm parameters are configured identically to Section 4 targeting 1% IGD optimality gap. For each topology, the IGD score is computed between the empirical Pareto front (M) produced by an algorithm and the true optimal front (R). This process repeats over 50 runs, yielding mean IGD and standard deviation. Table 2 compiles the IGD statistics across all trials for the NS-MF, MHNSGA and PSA algorithms. Proposed NS-MF achieves superior IGD scores compared to prior arts, substantiating convergence to true Pareto fronts. The results validate NS-MF's capabilities in reconciling diversity and proximity objectives when approximating optimal trade-off surfaces for controller placement.

6 Conclusion

This paper tackled the multi-objective optimization challenge of placing controllers in software-defined networks to balance performance and coordination costs. We modeled the NP-hard problem as reconciling switch-controller latency, failure tolerance, and inter-controller overhead. Our proposed NS-MF algorithm with tailored recombination and perturbation techniques can effectively approximate the Pareto frontier on large problem graphs. Extensive benchmarks against the exhaustive POCO solver demonstrated over 20X speedups for the biggest topologies with under 0.8% optimality gap. Detailed latency-controller count tradeoff characterization provided insights into topology-specific trends. Comparisons using the IGD metric substantiated NS-MF's capabilities in simultaneously achieving diversity and proximity objectives when estimating Pareto-optimal sets for resilient controller positioning. The method surpassed existing methaheuristics on some topologies. Although the current approach centers on propagation latencies, augmenting with dynamic traffic and failure patterns can enhance agility. Investigating alternative problem representations beyond facility locations may further enrich the multi-criteria optimization. Ultimately, the proposed NS-MF methodology and findings contribute efficient and high-quality approximations of the Pareto front to balance key aspects of robust SDN controller deployment.

Acknowledgements

The author has been supported by Gonbad Kavous University within the project with title "Investigating Quality-of-Service Metrics in Software Defined Networks" with grant no. 6/482.

References

[1] B. Almadani, A. Beg, and A. Mahmoud, *DSF: A distributed SDN control plane framework for the east/west interface*, IEEE Access **9** (2021), 26735–26754.

[2] X. Cai, Y. Xiao, M. Li, H. Hu, H. Ishibuchi, and X. Li, *A grid-based inverted generational distance for multi/many-objective optimization*, IEEE Trans. Evolut. Comput. **25** (2020), no. 1, 21–34.

[3] S. Dou, L. Qi, C. Yao, and Z. Guo, *Exploring the impact of critical programmability on controller placement for software-defined wide area networks*, IEEE/ACM Trans. Network. **31** (2023), no. 6, 2575–2588.

[4] S. Favuzza, M.G. Ippolito, and E.R. Sanseverino, *Crowded comparison operators for constraints handling in NSGA-II for optimal design of the compensation system in electrical distribution networks*, Adv. Engin. Inf. **20** (2006), no. 2, 201–211.

[5] D. Hock, S. Gebert, M. Hartmann, T. Zinner, and P. Tran-Gia, *POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks*, IEEE Network Oper. Manag. Symp., 2014, pp. 1–2.

[6] D. Hock, M. Hartmann, S. Gebert, T. Zinner, and P. Tran-Gia, *POCO-PLC: Enabling dynamic Pareto-optimal resilient controller placement in SDN networks*, IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS), IEEE, 2014, pp. 115–116.

[7] A.A. Ibrahim, F. Hashim, A. Sali, N.K. Noordin, K. Navaie, and S.M. Fadul, *Reliability-aware swarm based multi-objective optimization for controller placement in distributed SDN architecture*, Digital Commun. Networks **10** (2024), no. 5, 1245–1257.

[8] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, *Modified distance calculation in generational distance and inverted generational distance*, Evolut. Multi-Criterion Optim.: 8th Int. Conf., EMO 2015, Guimarães, Portugal, March 29–April 1, 2015, Springer International Publishing, Proc. Part II 8, 2015, pp. 110–125.

[9] B. Isong, R.R.S. Molose, A.M. Abu-Mahfouz, and N. Dladlu, Comprehensive review of SDN controller placement strategies. IEEE Access **8** (2020), 170070–170092.

[10] A. Jalili and M. Keshtgari, *A new reliable controller placement model for software-defined WANs*, J. AI Data Min. **8** (2020), no. 2, 269–277.

[11] A. Jalili, M. Keshtgari, and R. Akbari, *A new framework for reliable control placement in software-defined networks based on multi-criteria clustering approach*, Soft Comput. 24 (2020), no. 4, 2897–2916.

[12] R. Jeya, G.R. Venkatakrishnan, and V. Nagarajan, *Placing controllers using latency metrics in a smart grid implementing software-defined networking architecture*, Adv. Sci. Technol. **124** (2023), 828–835.

[13] K. Kaur, U. Singh, and R. Salgotra, *An enhanced moth flame optimization*, Neural Comput. Appl. **32** (2020), 2315–2349.

[14] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, *The internet topology zoo*, IEEE J. Selected Areas Commun. **29** (2011), no. 9, 1765–1775.

[15] Y. Li, S. Guan, C. Zhang, and W. Sun, *Parameter optimization model of heuristic algorithms for controller placement problem in large-scale SDN*, IEEE Access **8** (2020), 151668–151680.

[16] J. Ma, J. Chen, L. Dong, and X. Jiang, (2023). *Research on placement of distributed SDN multiple controllers based on IAVOA*, Cluster Comput. **27** (2024), no. 1, 913–930.

[17] Y. Maleh, Y. Qasmaoui, K. El Gholami, Y. Sadqi, and S. Mounir, *A comprehensive survey on SDN security: Threats, mitigations, and future directions*, J. Rel. Intel. Envir. **9** (2023), no. 2, 201–239.

[18] A. Naseri, M. Ahmadi, and L. PourKarimi, *Placement of SDN controllers based on network setup cost and latency of control packets*, Comput. Commun. **208** (2023), 15–28.

[19] Y.S.D. Phaneendra, U. Prabu, and S. Yasmine, *A study on multi-controller placement problem (MCPP) in software-defined networks*, Int. Conf. Sustain. Comput. Data Commun. Syst., IEEE, 2023, pp. 1454–1458.

[20] M.G. Resendel and C.C. Ribeiro, *GRASP with path-relinking: Recent advances and applications*, Metaheuristics: Progress as Real Problem Solvers, Springer, 2005, pp. 29–63.

[21] M. Shehab, L. Abualigah, H. Al Hamad, H. Alabool, M. Alshinwan, and A.M. Khasawneh, *Moth-flame optimization algorithm: Variants and applications*, Neural Comput. Appl. **32** (2020), 9859–9884.

[22] M. Shehab, H. Alshawabkah, L. Abualigah, and N. AL-Madi, *Enhanced a hybrid moth-flame optimization algorithm using new selection schemes*, Engin. Comput. **37** (2021), 2931–2956.

[23] T. Singh, N. Saxena, M. Khurana, D. Singh, M. Abdalla, and H. Alshazly, *Data clustering using moth-flame optimization algorithm*, Sensors **21** (2021), no. 12, 4086.

[24] X. Su, C. Zhang, C. Chen, L. Fang, and W. Ji, *Dynamic configuration method of flexible workshop resources based on IICA-NS algorithm*, Processes **10** (2022), no. 11, 2394.

[25] C. Xu, C. Xu, B. Li, S. Li, and T. Li, *Load-aware dynamic controller placement based on deep reinforcement learning in SDN-enabled mobile cloud-edge computing networks*, Comput. Networks **234** (2023), 109900.