

A Comprehensive Comparative Study of Metaheuristic Algorithms for the Controller Placement Problem in SDN-IoT Networks

Rami Mohaisen¹, Mohammad Muhaisen¹, Omar Fayez², Ayat M. Alrjoob³, Mamoun F. Al-Mistarihi¹, and Khalid A. Darabkh⁴

⁽¹⁾ Electrical Engineering Department, Jordan University of Science and Technology, P.O. Box 3030, Irbid 22110, Jordan
Emails: rfmoahaisen09@eng.just.edu.jo, mfmuhaisen11@eng.just.edu.jo, mistarihi@just.edu.jo

⁽²⁾ Electrical Engineering Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
Email: omar.20.3@hotmail.com

⁽³⁾ Electrical and Energy Engineering, School of Sustainable Systems Engineering, German Jordanian University, Amman 11180, Jordan
Email: ayat.alrjoob@gnu.edu.jo

⁽⁴⁾ Computer Engineering Department, School of Engineering, The University of Jordan, Amman 11942, Jordan
Email: k.darabkeh@ju.edu.jo

Abstract - The growth of Internet of Things (IoT) networks has increased the need for network management that can handle devices using Software-Defined Networking (SDN). A key challenge in SDN-IoT systems is where to place the controllers, known as the Controller Placement Problem (CPP). This is a task that requires finding the best locations for controllers to reduce delay, increase reliability, and lower costs. Although several solutions have been proposed for the CPP, they have not been compared under fair and identical conditions. This study compares four algorithms: Social Network Search (SNS), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and a combination of Differential Evolution and Whale Optimization Algorithm (DEWO). We tested these algorithms under identical conditions on IoT networks of different sizes, ranging from 50 to 1000 nodes. The results show that SNS achieves an average delay of 2.01 milliseconds across all network sizes and performs best overall. SNS is significantly better than GA. PSO is the fastest, taking 0.304 seconds to deploy a 1000-node network. These findings help researchers choose the most suitable algorithm for SDN-IoT deployments. The study provides a comparison that highlights the strengths and weaknesses of each algorithm and can guide future research in this area.

Keywords: Internet of Things (IoT), Software-Defined Networking (SDN), Controller Placement Problem (CPP), Social Network Search (SNS), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), Whale Optimization Algorithm (WOA), Benchmarking.

I. INTRODUCTION

The confluence of Software-Defined Networking (SDN) and the Internet of Things (IoT) has heralded new possibilities for agile, centrally managed control over heterogeneous device landscapes [1–7]. SDN disentangles the control plane from the data plane, allowing widespread control of vast IoT implementations via specialized controller nodes [8–9]. With this design comes an unavoidable tradeoff: the Controller Placement Problem (CPP), which seeks to identify the most efficient collection of network nodes to function as controllers [10–15].

The CPP is known to be NP-hard [16], implying that exact solutions are infeasible for large scales. Controller placement plays a crucial role in propagation latency, network reliability under device/link failures, infrastructure cost, and load balance over the control plane. These conflicting objectives need to be optimized together to manage SDN-IoT efficiently, especially in complex use cases like smart cities and industrial automation.

Several metaheuristic algorithms have been proposed for CPP, such as evolutionary [17], swarm intelligence [18], hybrid bio-inspired, and community-detection heuristics [19–22]. In our previous work [24], we showed that Social Network Search (SNS), which mimics people’s social interaction behavior, outperforms exhaustive optimum search. Unfortunately, there is an important missing piece: no side-by-side comparison of these metaheuristics has been performed on the same problem definition, exact networks, and testing conditions.

This paper addresses this gap by presenting the first comprehensive benchmarking study of SNS, GA [25], PSO [26], and DEWO [27] for the CPP. The main contributions are:

1. First systematic benchmarking under a unified fitness function.
2. Evaluation across eight network scales (from 50 to 1000 nodes).
3. Multi-metric analysis of latency, execution time, reliability, controller count, and convergence.
4. Statistical significance testing.

The remainder of the paper is organized as follows. Section II describes related work. Section III describes the system model and problem formulation. Section IV describes the four compared algorithms. Performance evaluation is described in Section V. Finally, the paper is concluded in Section VI

II. RELATED WORK

Following the formalization of SDN architectures, the CPP has gained major research interest. Mehraghdam et al. [16] proved the NP-hard complexity of the CPP, which motivated heuristic approaches for large-scale deployments. Sallahi and

St-Hilaire [10] proposed an ILP optimal model aiming to minimize the worst-case latency and offered a benchmark for comparing the heuristics.

Evolutionary approaches include the submodularity-based framework by Tran et al. [17] that considers coverage, update, connectivity, quality and budget requirements. Choumas et al. [18] used integer quadratic programming (IQP) formulations for energy efficiency in IoT systems. Ali and Roh [19] applied the Analytical Network Process with graph clustering to SDN-IoT smart cities. Keshari et al. [20] proposed DEWO, which combines Differential Evolution mutation with the spiral movement of Whale Optimization. Khera and Kurmi [21] improved cellular IoT networks using unsupervised machine learning integrated with Partition Around Medoids.

Community-structure approaches include Abderrahmane and Drid [22], who combined Louvain detection with betweenness-centrality metrics, noting limitations in controller count minimization and capacity constraints. Our prior work [24] introduced SNS for the CPP, demonstrating 119.6-second execution for 1000-node deployments versus computationally infeasible exact methods beyond 300 nodes. The present work extends [24] by providing the first systematic head-to-head comparison of SNS against three competing metaheuristics under identical conditions and distribution models within an improved evaluation environment. A critical gap that the surveyed literature has not addressed.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. Network Model

The SDN-IoT network is modelled as a weighted undirected graph $G(V, E, C)$, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of IoT nodes, $E = \{e_1, e_2, \dots, e_m\}$ models physical communication links with propagation delays, and $C \subseteq V$ identifies the deployed controller set. Edge weights represent propagation delays uniformly distributed in $[0.5, 3.0]$ ms. Network topologies are generated as Erdős-Rényi random graphs with edge probability $p = \max(0.03, 8/|V|)$, guaranteeing connectivity across all tested scales. Fig. 1 illustrates the considered SDN-IoT system model and use-case scenario. IoT nodes and their links form the data plane; a subset of nodes selected by the CPP solver hosts the SDN controllers, which form the control plane and communicate with the devices through the southbound interface. The four compared metaheuristics run in the application/management plane, where every candidate placement is scored by the unified fitness function of Section III.B and the resulting best placement is deployed. This scenario is representative of large-scale deployments such as smart cities and industrial automation, in which every IoT device must reach its nearest controller with minimal latency.

B. Unified Fitness Function

All four algorithms are evaluated using the identical multi-objective fitness function:

$$F(x) = w_1 L(x) + w_2 R(x) + w_3 C(x) + w_4 Q(x) \quad (1)$$

where x denotes the controller placement decision. $L(x)$ is the average shortest-path latency to the nearest controller; $R(x)$ characterizes the network's fault tolerance, including controller uptime, path redundancy, and failure resilience; $C(x) = |C|/|V|$ is the normalized controller count; and $Q(x)$ penalizes overloaded controllers. Weights $w_1 = 0.4$, $w_2 = w_3 = w_4 = 0.2$ prioritize latency minimization. All-pairs shortest paths are precomputed once per topology using Dijkstra's algorithm, enabling efficient repeated evaluations.

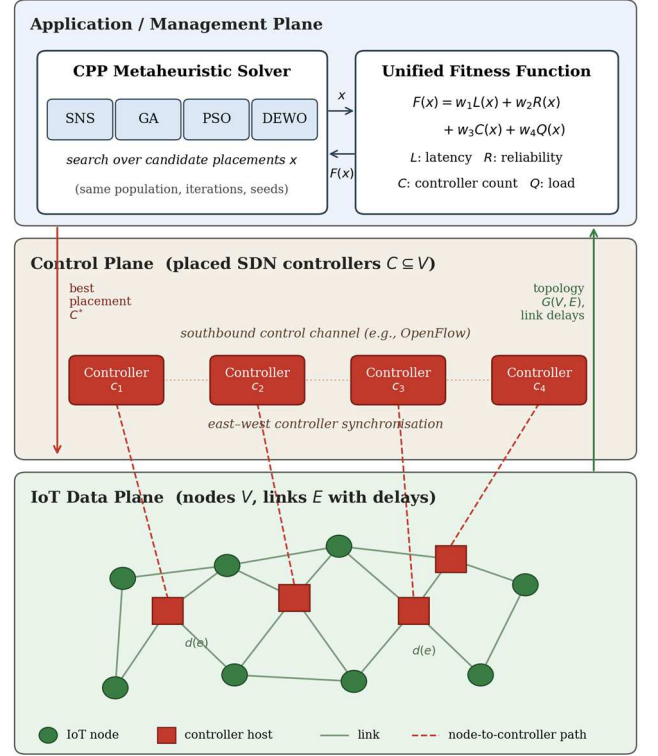


Fig. 1 System model of the SDN-IoT environment with the compared CPP solutions

IV. COMPARED ALGORITHMS

A. Social Network Search (SNS)

SNS [15, 24] models the dynamics of human social interaction. In each iteration, each agent updates its controller placement by one of four randomly selected behavioral modes: (1) Imitation: generating near copies of the current best solution, (2) Conversation: weighted combination of the best solution with a random candidate, (3) Disputation: combination of the inverse of a random solution with the best solution to simulate an adversarial dynamic, and (4) Innovation: exploration of uncharted regions by adding or deleting controllers. The probabilistic selection of the mode allows diverse exploration while taking advantage of the best-known configuration. The complete algorithm is shown in Algorithm 1.

B. Genetic Algorithm (GA)

GA [25] evolves a population of controller configurations through tournament selection (size 3), uniform crossover

combining the union of two parent sets, and bit-flip mutation (rate 0.15). Elitism preserves the best solution across generations. The variable-length chromosome encodes controller node indices, with crossover offspring sized at the mean of their parents. The complete algorithm is shown in Algorithm 2.

Algorithm 1: Social Network Search (SNS) Optimization

```

1: Input: Number of nodes  $N$ , Max iterations  $\text{Max}_{\text{iter}}$ , Problem parameters
2: Output: Optimal solution
3: Initialize:
4:   Initialize population of solutions randomly
5:   Evaluate fitness of initial solutions
6:   Identify best solution  $S_{\text{best}}$ 
7:    $\text{iter} \leftarrow 0$ 
8: while  $\text{iter} < \text{Max}_{\text{iter}}$  do
9:    $\text{iter} \leftarrow \text{iter} + 1$ 
10:  for each current solution  $S_i$  in population do
11:     $\text{mode} \leftarrow$  Randomly select integer from  $\{1, 2, 3, 4\}$ 
12:    if  $\text{mode} = 1$  then
13:       $S_{\text{new}} \leftarrow \text{Imitation}(S_i) \triangleright$  Generate new solution using Imitation mode
14:    else if  $\text{mode} = 2$  then
15:       $S_{\text{new}} \leftarrow \text{Conversation}(S_i) \triangleright$  Generate new solution using Conversation mode
16:    else if  $\text{mode} = 3$  then
17:       $S_{\text{new}} \leftarrow \text{Disputation}(S_i) \triangleright$  Generate new solution using Disputation mode
18:    else if  $\text{mode} = 4$  then
19:       $S_{\text{new}} \leftarrow \text{Innovation}(S_i) \triangleright$  Generate new solution using Innovation mode
20:    end if
21:    if  $\text{Fitness}(S_{\text{new}}) > \text{Fitness}(S_i)$  then
22:      Replace  $S_i$  with  $S_{\text{new}} \triangleright$  Keep the better solution
23:    else
24:      Keep  $S_i \triangleright$  Discard new solution
25:    end if
26:    if  $\text{Fitness}(S_i) > \text{Fitness}(S_{\text{best}})$  then
27:      Update  $S_{\text{best}} \leftarrow S_i$ 
28:    end if
29:  end for
30: end while
31: Return:  $S_{\text{best}}$ 

```

C. Particle Swarm Optimization (PSO)

PSO [26] maintains a swarm of particles, each encoding a continuous probability vector over all network nodes. Controller selection is performed using a threshold of $p > 0.5$, with top- k nodes by probability assigned as controllers. Position updates follow standard PSO dynamics: inertia weight $w = 0.729$, cognitive and social acceleration coefficients $c_1 = c_2 = 1.494$, balancing exploration and exploitation throughout the search. The complete algorithm is shown in Algorithm 3.

D. Differential Evolution with Whale Optimization (DEWO)

DEWO [27] hybridizes Differential Evolution (scale factor $F = 0.5$, crossover rate $CR = 0.7$) with the Whale Optimization Algorithm's spiral and shrinking encircling mechanisms [28]. At each iteration, agents probabilistically select between a DE trial vector and a WOA position update, combining DE's directed mutation with WOA's nonlinear spiral exploration for enhanced solution space coverage. The complete algorithm is shown in Algorithm 4.

Algorithm 2: Genetic Algorithm (GA) Optimization

```

1: Input: Number of nodes  $N$ ,  $\text{Max}_{\text{iter}}$ ,  $\text{Pop}_{\text{size}}$ , mutation rate  $\mu = 0.15$ 
2: Output: Optimal solution  $S_{\text{best}}$ 
3: Initialize:
4:   Initialize population  $\{S_1, \dots, S_{\text{Pop}_{\text{size}}}\}$  with random controller sets
5:   Evaluate fitness of all chromosomes
6:   Identify best solution  $S_{\text{best}}$ 
7:    $\text{iter} \leftarrow 0$ 
8: while  $\text{iter} < \text{Max}_{\text{iter}}$  do
9:    $\text{iter} \leftarrow \text{iter} + 1$ 
10:   $\text{new\_pop} \leftarrow \{S_{\text{best}}\} \triangleright$  Elitism: preserve best solution
11:  while  $|\text{new\_pop}| < \text{Pop}_{\text{size}}$  do
12:     $P_1 \leftarrow \text{Tournament\_Select}(\text{population}) \triangleright$  Tournament size = 3
13:     $P_2 \leftarrow \text{Tournament\_Select}(\text{population}) \triangleright$  Tournament size = 3
14:     $S_{\text{child}} \leftarrow \text{Crossover}(P_1, P_2) \triangleright$  Union-based crossover
15:     $S_{\text{child}} \leftarrow \text{Mutate}(S_{\text{child}}, \mu) \triangleright$  Random node replacement
16:     $\text{new\_pop} \leftarrow \text{new\_pop} \cup \{S_{\text{child}}\}$ 
17:  end while
18:   $\text{population} \leftarrow \text{new\_pop}$ 
19:  Evaluate fitness of new population
20:   $S_{\text{iter}} \leftarrow \arg \min F(S_i)$  for all  $S_i \in \text{population}$ 
21:  if  $\text{Fitness}(S_{\text{iter}}) > \text{Fitness}(S_{\text{best}})$  then
22:    Update  $S_{\text{best}} \leftarrow S_{\text{iter}}$ 
23:  end if
24: end while
25: Return:  $S_{\text{best}}$ 

```

V. PERFORMANCE EVALUATION

A. Experimental Setup

Experiments use eight Erdős-Rényi IoT topologies: $|V| \in \{50, 100, 150, 200, 300, 500, 700, 1000\}$ nodes. Population sizes scale from 20 (≤ 300 nodes) to 10 (1000 nodes); iteration counts from 100 to 35, respectively, to maintain feasibility at scale. Each configuration is independently repeated several times with different random seeds; reported values are means. Experiments are implemented in Python 3 using NumPy, NetworkX, multiprocessing, and SciPy on a dual Intel Xeon workstation. Several enhancements were made to the original code to improve execution performance, such as the exploitation of parallelism and modifications to loop blocks to reduce redundant executions. Consequently, the results differ from those reported in our original paper [24].

B. Execution Time

As shown in Table I and Fig. 2, PSO achieves the fastest execution times across all scales due to its fully vectorized position updates. PSO completes 1000-node deployments in 0.304 s, versus 0.354 s (SNS), 0.349 s (DEWO), and 0.374 s (GA). All four algorithms complete the largest tested topology (1000 nodes) in well under one second (0.30–0.37 s), as reported in Table I; the execution times therefore lie in the sub-second range across all scales. GA exhibits the highest overhead due to list-based chromosome operations. PSO's speed advantage over SNS is statistically significant ($p = 0.002$, Table II). The difference in execution time compared to our previous results [24] arises because we used the multiprocessing library in our Python code, which made the

execution time significantly faster by leveraging multiprocessor concurrent execution.

Table I: Execution Time Comparison (Seconds)

Nodes	SNS	GA	PSO	DEWO
50	0.21	0.25	0.13	0.22
100	0.25	0.29	0.17	0.29
200	0.42	0.45	0.26	0.39
300	0.54	0.58	0.33	0.48
500	0.47	0.50	0.34	0.42
700	0.38	0.42	0.31	0.36
1000	0.35	0.37	0.30	0.35

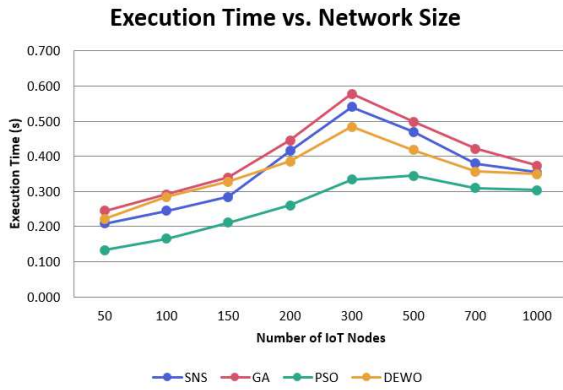


Fig. 2 Execution time vs. network size

Algorithm 3: Particle Swarm Optimisation (PSO)

```

1: Input: Number of nodes  $N$ ,  $\text{Max}_{\text{iter}}$ ,  $\text{Pop}_{\text{size}}$ ,  $w=0.729$ ,  $c_1=c_2=1.494$ 
2: Output: Optimal solution  $S_{\text{best}}$ 
3: Initialize:
4:   Initialize  $\mathbf{X} \in \mathbb{R}^{\text{Pop} \times N}$ ,  $\mathbf{V} \in \mathbb{R}^{\text{Pop} \times N}$  randomly in  $[0, 1]$ 
5:   Decode each  $X_i$  to controller set  $S_i \triangleright$  Top- $k$  nodes by probability
6:   Evaluate fitness; set  $P_{\text{best}} \leftarrow \mathbf{X} \triangleright$  Personal best positions
7:    $G_{\text{best}} \leftarrow \arg \min F(\text{Decode}(X_i)) \forall i \triangleright$  Global best position
8:    $\text{iter} \leftarrow 0$ 
9: while  $\text{iter} < \text{Max}_{\text{iter}}$  do
10:   $\text{iter} \leftarrow \text{iter} + 1$ 
11:  for each particle  $i$  in swarm do
12:     $r_1, r_2 \leftarrow \text{Uniform}(0, 1)$ 
13:     $V_i \leftarrow w \cdot V_i + c_1 \cdot r_1 \cdot (P_{\text{best},i} - X_i) + c_2 \cdot r_2 \cdot (G_{\text{best}} - X_i)$ 
14:     $V_i \leftarrow \text{clip}(V_i, -6, 6) \triangleright$  Velocity clamping
15:     $X_i \leftarrow \text{clip}(X_i + V_i, 0, 1) \triangleright$  Position update
16:     $S_{\text{new}} \leftarrow \text{Decode}(X_i) \triangleright$  Select top- $k$  nodes as controllers
17:    if  $\text{Fitness}(S_{\text{new}}) > \text{Fitness}(\text{Decode}(P_{\text{best},i}))$  then
18:       $P_{\text{best},i} \leftarrow X_i \triangleright$  Update personal best
19:      if  $\text{Fitness}(S_{\text{new}}) > \text{Fitness}(\text{Decode}(G_{\text{best}}))$  then
20:         $G_{\text{best}} \leftarrow X_i \triangleright$  Update global best
21:      end if
22:    end if
23:  end for
24: end while
25: Return:  $\text{Decode}(G_{\text{best}})$ 

```

Algorithm 4: Differential Evolution with Whale Optimisation (DEWO)

```

1: Input: Number of nodes  $N$ ,  $\text{Max}_{\text{iter}}$ ,  $\text{Pop}_{\text{size}}$ ,  $F=0.5$ ,  $\text{CR}=0.7$ ,  $b=1.0$ 
2: Output: Optimal solution  $S_{\text{best}}$ 
3: Initialize:
4:   Initialize position  $\mathbf{X} \in \mathbb{R}^{\text{Pop} \times N}$  randomly in  $[0, 1]$ 
5:   Evaluate fitness; identify best position  $X_{\text{best}}$ 
6:    $\text{iter} \leftarrow 0$ 
7: while  $\text{iter} < \text{Max}_{\text{iter}}$  do
8:    $\text{iter} \leftarrow \text{iter} + 1$ 
9:    $a \leftarrow 2 - 2 \cdot (\text{iter} / \text{Max}_{\text{iter}}) \triangleright$  Linearly decreasing control factor
10:  for each agent  $i$  in population do
11:    if  $\text{rand}() < 0.5$  then  $\triangleright$  Differential Evolution (DE) step
12:      Select  $r_1, r_2, r_3$  from population,  $r_1 \neq i$ 
13:       $X_{\text{mut}} \leftarrow \text{clip}(X_{r_1} + F \cdot (X_{r_2} - X_{r_3}), 0, 1) \triangleright$  DE mutation
14:       $X_{\text{trial}[i]} \leftarrow X_{\text{mut}[i]}$  if  $\text{rand}() < \text{CR}$  else  $X_{\text{trial}[i]} \leftarrow X_{\text{best}[i]}$   $\triangleright$  Crossover
15:    else  $\triangleright$  Whale Optimisation Algorithm (WOA) step
16:      if  $\text{rand}() < 0.5$  then  $\triangleright$  Shrinking encircling mechanism
17:         $A \leftarrow 2a \cdot \text{rand}() - a$ ;  $D \leftarrow |2 \cdot \text{rand}() \cdot X_{\text{best}} - X_i|$ 
18:         $X_{\text{trial}} \leftarrow \text{clip}(X_{\text{best}} - A \cdot D, 0, 1)$ 
19:      else  $\triangleright$  Spiral bubble-net update
20:         $l \leftarrow \text{Uniform}(-1, 1)$ ;  $D \leftarrow |X_{\text{best}} - X_i|$ 
21:         $X_{\text{trial}} \leftarrow \text{clip}(D \cdot e^{b \cdot l} \cdot \cos(2\pi l) + X_{\text{best}}, 0, 1)$ 
22:      end if
23:    end if
24:    if  $\text{Fitness}(\text{Decode}(X_{\text{trial}})) > \text{Fitness}(\text{Decode}(X_i))$  then
25:       $X_i \leftarrow X_{\text{trial}} \triangleright$  Greedy selection
26:    if  $\text{Fitness}(\text{Decode}(X_i)) > \text{Fitness}(\text{Decode}(X_{\text{best}}))$  then
27:       $X_{\text{best}} \leftarrow X_i \triangleright$  Update best position
28:    end if
29:  end for
30: end while
31: Return:  $\text{Decode}(X_{\text{best}})$ 

```

C. Average End-to-End Latency

Fig. 3 plots latency versus network size. SNS shows the lowest latency across all tested scales (mean overall: 2.01 ms). DEWO (2.1 ms) and PSO (2.069 ms) show similar intermediate performance, while GA consistently has the highest latency (2.116 ms). One interesting trend is the decrease in latency with increasing network size. Larger topologies allow a finer-grained placement of controllers and hence shorter average node-to-controller distances. For 1000 nodes, SNS achieves 1.800 ms while GA achieves 1.839 ms. The superiority of SNS over GA is statistically significant ($p = 0.030$, Table II).

Average End-to-End Latency vs. Network Size

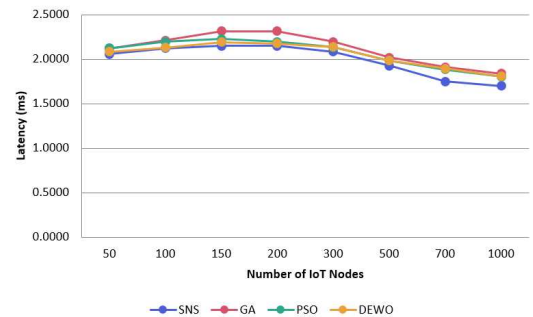


Fig. 3 Average end-to-end latency vs. network size. SNS consistently achieves the lowest latency

D. Network Reliability

All four algorithms achieve about 90% reliability for all eight topologies tested (Fig. 4), ensuring that almost every IoT node has a path to controllers. The uniformity of results shows the robustness of the metaheuristic controller placement, but also points out the lack of sufficient differentiation in this metric in the tested Erdős-Rényi topologies with relatively high edge density. Future work should examine reliability under sparser topologies and explicit link failure scenarios to better discriminate algorithm performance on this dimension.

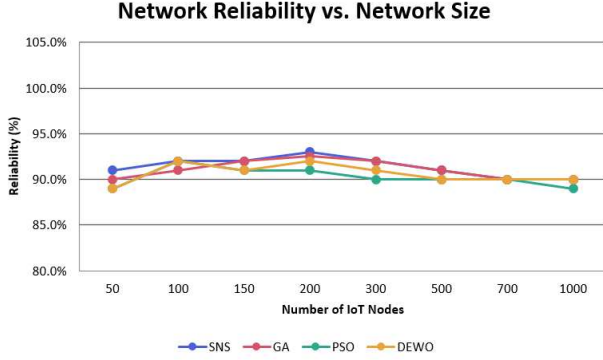


Fig. 4 Network reliability (%) vs. network size. All algorithms achieve around 90% across all scales

E. Controller Count

Fig. 5 shows the number of controllers deployed. PSO and DEWO utilize the same search bounds, which is why they returned identical controller counts at all scales. GA and SNS consistently use fewer controllers (e.g., 113 and 110, respectively, versus 125 at 1000 nodes). For GA, however, the reduced number of controllers comes at the expense of increased latency and lower global fitness, which is an undesirable trade-off in most deployment scenarios, where Quality of Service is of primary concern.

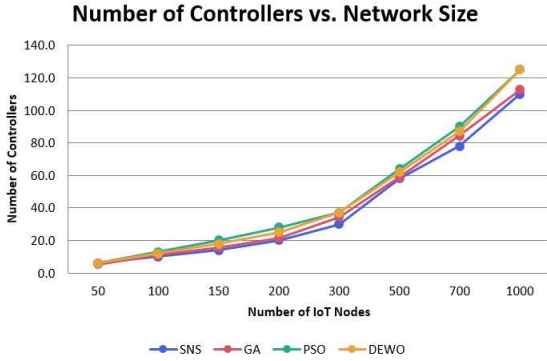


Fig. 5 Number of deployed controllers vs. network size

F. Convergence Analysis

The convergence curves for three typical scales are plotted in Figs. 6, 7 and 8. Both SNS and DEWO demonstrate rapid initial gain and then gradual refinement, suggesting structured exploitation of the current best solution. PSO converges fast on early iterations but exhibits a plateau effect in large search spaces, which shows a premature convergence. GA has the slowest convergence because of the genetic drift in the early

evolution. The profiles corroborate the balanced exploration–exploitation dynamics of SNS as a structural advantage of its four-mode framework.

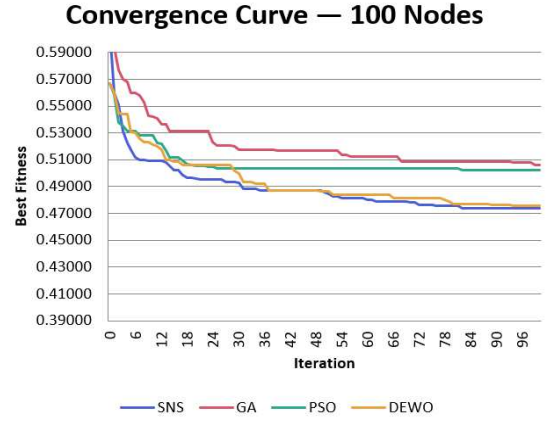


Fig. 6 Convergence curves at 100 nodes

G. Statistical Analysis

The results of the Wilcoxon Rank-Sum test for all eight network scales are shown in Table II. Significance codes: $p < 0.01$, $p < 0.05$, ns = not significant. Statistically, SNS is significantly better than GA in latency ($p = 0.030$) and fitness ($p = 0.035$). The differences between SNS and DEWO or PSO consistently favor SNS but do not reach statistical significance within the tested scale range, indicating competitive performance. The advantage of PSO over SNS in execution time is statistically significant ($p = 0.002$).

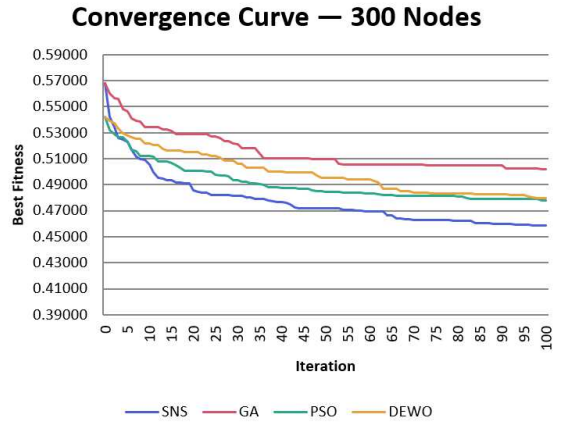


Fig. 7 Convergence curves at 300 nodes

VI. CONCLUSION

This paper is the first systematic benchmarking of four metaheuristic algorithms, SNS, GA, PSO, and DEWO, for the CPP in SDN-IoT networks under a unified experimental setting. We experiment on eight network scales (from 50 to 1000 nodes) and confirm that SNS achieves the best performance in terms of latency (mean 2.01 ms) and composite fitness and outperforms GA significantly ($p < 0.05$). PSO has the best execution efficiency (0.304 s with 1000 nodes), while DEWO provides reliable second-best solution quality. All algorithms achieve around 90% reliability at the tested distribution and density of the networks. These results provide a useful benchmark for

algorithm selection in SDN-IoT deployments: SNS if solution quality is prioritized, PSO if execution speed is prioritized, and DEWO as a reliable quality-speed compromise. Future directions include energy-aware fitness functions incorporating deployment costs, dynamic topology adaptation for mobile IoT, multi-domain deployments, evaluation under fault-prone sparse topologies, and integration of reinforcement learning for real-time controller replacement in evolving IoT ecosystems.

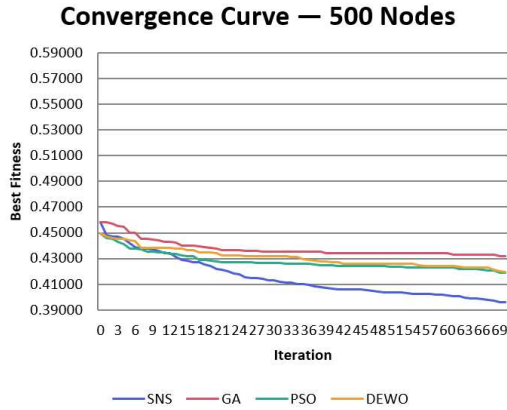


Fig. 8 Convergence curves at 500 nodes

Table II: Statistical Significance - Wilcoxon Rank-Sum Tests

Metric	Comparison	p-value	Sig.	Winner
Latency (ms)	SNS vs GA	0.030	*	SNS
Latency (ms)	SNS vs PSO	0.080	ns	SNS
Latency (ms)	SNS vs DEWO	0.257	ns	SNS
Fitness F(x)	SNS vs GA	0.035	*	SNS
Fitness F(x)	SNS vs PSO	0.083	ns	SNS
Exec. Time	SNS vs PSO	0.002	**	PSO

REFERENCES

- [1] Darabkh, K. A., Al-Mistarihi, M. F., and Al-Maaitah, M. I., "Next-generation routing for autonomous vehicle networks based on innovative clustering: integrating SDN and fog computing along with AODV upon failure," *Journal of Supercomputing*, Springer, vol. 81, no. 2, p. 379, DOI: <http://doi.org/10.1007/s11227-024-06880-6>, 2025.
- [2] Vukovic, A., & Alrjoob, A. (2020, September). Furthering innovation in hyper communication era. In *2020 Global Congress on Electrical Engineering (GC-ElecEng)* (pp. 1-5). IEEE.
- [3] Darabkh, K. A., Al-Mistarihi, M. F., and Al-Maaitah, M. I., "A Yet Powerful Routing Protocol for Vehicular Ad-hoc Networks (VANETs) Utilizing Software Defined Network (SDN) and Fog Computing," *Proc. of the 8th International Conference on Information Technology, Information Systems and Electrical Engineering, ICITISEE-2024*, Grand Keisha Yogyakarta, Indonesia, August 29-30, DOI: <http://doi.org/10.1109/ICITISEE63424.2024.10730207>, 2024.
- [4] Darabkh, K. A., et al., "ICDRP-F-SDVN: An Innovative Cluster-Based Dual-Phase Routing Protocol Using Fog Computing and SDN," *Vehicular Communications*, vol. 34, p. 100453, Apr. 2022.
- [5] Al-Shurman, M., Al-Mistarihi, M. F., and Qudaimat, A., "Network Address Assignment in Mobile Ad-Hoc Networks," *Proc. International Congress on Ultra Modern Telecommunications and Control Systems, ICUMT2010*, Moscow, Russia, pp. 287 – 294, October 18-20, DOI: <https://doi.org/10.1109/ICUMT.2010.5676624>, 2010.
- [6] Alrjoob, A. M., Ababnah, A. A., Al-Mistarihi, M. F., and Darabkh, K. A., "Securing Primary Users in IoT 5G Cognitive Radio Networks," *Proc. of*

- the 32nd International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2024*, Bol (island of Brac), Split, Croatia, September 26-28, DOI:<http://doi.org/10.23919/SoftCOM62040.2024.10721968>, 2024.
- [7] Darabkh, K. A., Al-Mistarihi, M. F., and Odat, B. A., "Leveraging fog computing and software-defined networking for a novel velocity-aware routing protocol with election and handover thresholds in VANETs," *Journal of Supercomputing*, Springer, vol. 81, no. 2, p. 426, DOI: <http://doi.org/10.1007/s11227-024-06883-3>, 2025.
- [8] Darabkh, K. A., Al-Mistarihi, M. F., and Odat, B. A., "Integrating Software Defined Network and Fog Computing for a Novel Routing Protocol in Vehicular Ad-Hoc Networks," *Proc. of the 7th International Conference on Advanced Communication Technologies and Networking (CommNet 2024)*, Rabat, Morocco, December 4-6, DOI: <http://doi.org/10.1109/CommNet63022.2024.10793294>, 2024.
- [9] Darabkh, K. A., and Alkhader, B. Z., "Fog Computing- and SDN-Based Routing Protocol for VANETs," in *Proc. IEEE ICOIN 2022*, Jeju Island, Korea, January 2022.
- [10] Sallahi, A., and St-Hilaire, M., "Optimal model for the controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 1, pp. 30–33, 2015.
- [11] Luo, T., Tan, H.-P., and Quek, T. Q. S., "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012.
- [12] Hu, P., et al., "Security and privacy preservation scheme of face identification and resolution framework using fog computing in IoT," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1143–1155, Oct. 2017.
- [13] Ahmed, R., and Boutaba, R., "Design considerations for managing wide area software defined networks," *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 116–123, Jul. 2014.
- [14] Abderrahmane, A., Drid, H., and Behaz, A., "A survey of controller placement problem in SDN-IoT network," *Int. J. Networked Distrib. Comput.*, pp. 1–15, Jul. 2024.
- [15] Talatahari, S., Bayzidi, H., and Saraee, M., "Social network search for global optimization," *IEEE Access*, vol. 9, pp. 92815–92863, 2021.
- [16] Mehraghdam, S., Keller, M., and Karl, H., "Specifying and placing chains of virtual network functions," in *Proc. IEEE CloudNet*, Oct. 2014, pp. 7–13.
- [17] Tran, A. K., Piran, M. J., and Pham, C., "SDN controller placement in IoT networks: An optimized submodularity-based approach," *Sensors*, vol. 19, no. 24, p. 5474, Dec. 2019.
- [18] Choumas, K., Giatsios, D., Flegkas, P., and Korakis, T., "SDN controller placement and switch assignment for low power IoT," *Electronics*, vol. 9, no. 2, p. 325, Feb. 2020.
- [19] Ali, J., and Roh, B.-H., "An effective approach for controller placement in SD-IoT," *Sensors*, vol. 22, no. 8, p. 2992, Apr. 2022.
- [20] Keshari, S. K., Kansal, V., and Kumar, S., "An intelligent way for optimal controller placements in SDN-IoT networks for smart cities," *Comput. Ind. Eng.*, vol. 162, Art. no. 107667, Dec. 2021.
- [21] Khera, A., and Kurmi, U. S., "Enhancing performance of wide area CIoT SDN by US-ML based optimum controller placement," *Res. Rep. Comput. Sci.*, pp. 112–121, Jun. 2023.
- [22] Abderrahmane, A., and Drid, H., "Enhancing control placement in SDN-IoT networks using the Louvain algorithm and betweenness-centrality," *IEEE Access*, vol. 12, pp. 159775–159783, 2024.
- [23] Jupyter. [Online]. Available: <https://jupyter.org/>
- [24] Mohaisen, R., Muhaisen, M., Fageez, O., Alrjoob, A. M., Al-Mistarihi, M. F., and Darabkh, K. A., "Investigating Social Network Search (SNS) for the Control Placement Problem (CPP) in SDN-IoT Networks," *Proc. of IEEE 2026 International Black Sea Conference on Communications and Networking*, Bucharest, Romania, June 2026.
- [25] Holland, J. H., *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.
- [26] Kennedy, J., and Eberhart, R., "Particle swarm optimization," in *Proc. IEEE ICNN*, vol. 4, Perth, WA, Australia, Nov. 1995, pp. 1942–1948.
- [27] Storn, R., and Price, K., "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *J. Global Optim.*, vol. 11, pp. 341–359, 1997.
- [28] Mirjalili, S., and Lewis, A., "The Whale Optimization Algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.