2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA)

An Improved Particle Swarm Optimization Algorithm with Crossover and Mutation for Edge Deployment Optimization in Computational Networks

Zhanlin Liang
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
liang2018@bupt.edu.cn

Yunxiao Zu*
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
zuyx@bupt.edu.cn
*Corresponding author

Bin Hou*
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
robinhou@bupt.edu.cn
*Corresponding author

Mengying Xu
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
xmy222@bupt.edu.cn

Jing Chen
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
jingchen@bupt.edu.cn

Meiru Liu
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
meiruliu195@bupt.edu.cn

***Abstract*—Computational networks are increasingly relying on edge service deployments to meet the diverse demands of modern applications, where factors like latency, resource utilization, and deployment costs are critical. However, the heterogeneous and distributed nature of computational resources poses significant challenges in achieving efficient deployment. This paper presents a multi-objective optimization framework based on an improved particle swarm optimization (PSO) algorithm to address these challenges. The framework optimally allocates edge resources by considering key factors such as computational capacity, inter-node communication delay, user-to-node latency, and deployment costs. The proposed algorithm incorporates adaptive inertia weight adjustments and enhanced neighborhood exploration mechanisms to achieve better convergence and solution quality. The performance of the improved PSO framework, evaluated in terms of deployment cost, service latency, and resource utilization, is benchmarked against traditional optimization methods, including standard PSO and genetic algorithms (GA). Simulation results demonstrate that the proposed framework significantly outperforms existing methods, achieving lower latency and improved resource efficiency while reducing overall deployment costs.**

Keywords—Computing network, resource optimization, latency minimization, dynamic PSO, swarm intelligence

I. Introduction

Computational networks are becoming the backbone of modern digital infrastructure, driving innovations in edge computing, Internet of Things (IoT), and artificial intelligence. As one of the key paradigms in these networks, edge computing emphasizes bringing computation and data storage closer to the location of use to reduce latency and enhance real-time processing capabilities[1]. This paradigm shift is particularly important in latency-sensitive applications such as autonomous driving, augmented reality, and industrial automation, where the dynamic nature of data and stringent Quality of Service (QoS) requirements present significant challenges.

Edge service deployment plays a pivotal role in ensuring the efficient utilization of computational resources in these networks. By dynamically allocating services to edge nodes, computational networks can balance workloads, optimize latency, and minimize deployment costs[2]. However, the inherent heterogeneity of computational resources and the complex interdependencies between service placement, resource utilization, and communication delays pose significant challenges for effective deployment.

One of the critical challenges in edge service deployment is the development of multi-objective optimization strategies that simultaneously address latency, deployment cost, and resource efficiency. Traditional optimization methods, including static algorithms and heuristic approaches, often struggle to balance these objectives due to the dynamic and distributed nature of computational networks[3]. Furthermore, the increasing diversity of edge resources, coupled with user-specific QoS requirements, necessitates the consideration of fine-grained factors such as computational capacity, inter-node communication delays, and node-to-user latency.

To address these challenges, this paper introduces a nature-inspired multi-objective optimization framework for edge service deployment, leveraging an improved particle swarm optimization (PSO) algorithm. The contributions of this work are as follows:

(1) A novel multi-objective optimization model is proposed, which comprehensively considers factors such as computational capacity, inter-node latency, user-to-node latency, and deployment costs, ensuring practical applicability in real-world computational networks.

(2) An improved particle swarm optimization algorithm is designed with adaptive inertia weights and enhanced

neighborhood exploration, striking an effective balance between global exploration and local exploitation.

(3) A comprehensive performance evaluation is conducted, benchmarking the proposed approach against traditional optimization methods, including standard PSO and genetic algorithms (GA). Results demonstrate significant improvements in terms of latency reduction, cost efficiency, and resource utilization.

This study highlights the potential of nature-inspired algorithms in solving complex multi-objective optimization problems in edge computing, paving the way for more efficient and scalable computational network designs.

II. Related Work

The optimization of edge service deployment has been a sig nificant research focus in computational networks, with numero us studies exploring resource allocation, latency reduction, and cost efficiency. Traditional methods, heuristic algorithms, and n ature-inspired techniques have all contributed to this domain.

One of the pioneering approaches in this field involves deter ministic optimization. Chen et al. [4] proposed a computation o ffloading model for mobile edge computing in small-cell netwo rks. Their method effectively minimized energy consumption a nd latency in static network environments. However, the static nature of the approach made it less adaptable to the dynamic an d distributed characteristics of computational networks.

In heuristic methods, researchers have explored Particle Sw arm Optimization (PSO)-based strategies. Chafi et al. [5] devel oped a PSO-based algorithm to optimize workflow time and en ergy consumption in fog computing environments. Their results demonstrated enhanced resource allocation efficiency but requi red significant computation overhead for large-scale networks. Similarly, Abedifar et al. [6] utilized PSO to optimize virtual ne twork mapping in cloud computing, emphasizing its ability to a ddress resource constraints in virtualized environments. Vadive l et al. [7] further proposed a hybrid PSO-MGA (Modified Gen etic Algorithm) approach, enhancing PSO's global search capab ilities with genetic refinement techniques for demand-based res ource allocation. Despite the improvements, these methods ofte n faced challenges such as high convergence time and paramete r tuning.

Hybrid optimization strategies combining PSO with other m etaheuristics have gained attention due to their potential to bala nce exploration and exploitation. Alhaizaey et al. [8] introduced a PSO-based metaheuristic for task allocation in edge computi ng micro-clusters, addressing the heterogeneity of tasks and res ources. This approach provided a robust allocation mechanism but required additional computational resources for large-scale deployments. Singh and Kim [9] applied evolutionary algorithm s to jointly optimize computation offloading and resource alloca tion in cloud-RAN environments, achieving notable improveme nts in both performance metrics and scalability.

In addition to heuristic approaches, multi-objective optimiza tion frameworks have also been proposed to address the comple x trade-offs in edge service deployment. Wang et al. [10] prese nted a joint energy minimization and resource allocation model for cloud-RAN, achieving energy-efficient task assignment whi le considering user demands. Midya et al. [11] proposed a three -tier cloud architecture utilizing PSO to optimize resource alloc ation for VANETs, demonstrating significant improvements in latency reduction and energy conservation.

These studies highlight the diverse methodologies adopted i n edge service deployment. Deterministic approaches offer stab le solutions but often lack flexibility in dynamic environments. Heuristic and nature-inspired algorithms, particularly PSO and i ts hybrids, have demonstrated superior adaptability and perform ance in complex optimization scenarios. However, challenges s uch as computational complexity, parameter sensitivity, and sca lability remain. This paper builds upon these foundations by int roducing an improved PSO framework that addresses the limita tions of existing methods and achieves efficient resource allocat ion and latency optimization in computational networks.

III. System Model

In this section, a resource-aware deployment model is designed to optimize the placement of computational tasks in a computational network, with the primary objective of minimizing total cost and maximizing resource utilization. The proposed model considers three key metrics for task allocation: computational capacity of the nodes, latency between nodes and users, and deployment cost. These factors are incorporated into a multi-objective optimization framework, which guides the assignment of tasks to the most suitable computational nodes, ensuring efficiency and scalability in the network.

A. Network Model

The computational network is modeled as a directed graph $G=(S,U,E)$,where S,U ,and E denote the sets of computa tional nodes, tasks, and communication links, respectively. Spe cifically:

$S=\{s_1,s_2,...,s_N\}$ represents computational nodes, such as edge servers or cloud resources, each characterized by comp utational capacity, resource type, deployment cost, and energy consumption.

$U=\{u_1,u_2,...,u_M\}$ denotes tasks or user requests, each defined by specific resource demands, task type compatibility, and latency requirements.

E captures the communication links, where D_{ij}^{node} represe nts the latency between nodes s_i and s_j, and D_{iu}^{user} represent s the latency between node s_i and task u Each computational node s_i is characterized by the following parameters:

(1) Computational Capacity (C_i): Maximum computat ional workload the node can handle.

(2) Resource Type (P_i): The type of computational resou rces available (e.g., CPU, GPU, or FPGA).

(3) Deployment Cost (Y_i): The monetary cost of utilizing the node for task execution.

(4) Bandwidth (B_i): Maximum data transfer capacity sup ported by the node.

(5) Energy Consumption (E_i): Energy usage per task ex ecution. Each task u is described by:

(6) Computational Demand (T_u): The required resources for task execution.

(7) Task Type (T_u^{type}): The compatibility requirement for task execution on specific node types.

(8) Data Transmission Demand (β_u): The amount of dat a required to complete the task.

(9) Priority Level (P_u): The importance of the task in the scheduling process.

(10) Latency Tolerance ($D_u^{\max}$): The maximum tolerable l atency for task completion.

To optimize the system performance, a multi-objective fram ework is formulated. The overall objective function is expresse d as:

$$\min_x \sqrt{\sum_{i=1}^{5} \omega_i \bullet f_i(x)^2} \tag{1}$$

where $\omega_1, \omega_2, \omega_3, \omega_4, \omega_5$ are the weights assigned to indi vidual objectives.

B. *Sub-objective Functions*

In our computing network model, multiple sub-objective functions are designed to optimize different aspects of resource allocation and task scheduling. These include:

(1) Latency Minimization($f_1(x)$)

The latency minimization objective ensures tasks are deployed to nodes with lower communication delays:

$$f_1(x) = \sum_{u\in U} \alpha_u \bullet \sum_{i\in S} x_{ui} \bullet (\sum_{j\in S} x_{uj} \bullet D_{ij}^{node} \bullet \beta_u + D_{iu}^{user} \bullet \beta_u) \tag{2}$$

Where α_u represents the priority weight of task u and β_u reflects the data transmission volume associated with the task.

(2) Resource Utilization Optimization($f_2(x)$)

Resource utilization is optimized by balancing the workload across nodes:

$$f_2(x) = \sum_{i\in S} \left(\left| \frac{\sum_{u\in U} x_{ui} \bullet T_u}{C_i} - \eta \right| + \lambda \bullet \max(0, \frac{\sum_{u\in U} x_{ui} \bullet T_u}{C_i} - 1) \right) \tag{3}$$

where η denotes the ideal utilization rate (e.g., 80%), and λ penalizes node overload.

(3) Deployment Cost Minimization($f_3(x)$)

The cost of task deployment is minimized to reduce operational expenses:

$$f_3(x) = \sum_{u\in U} \sum_{i\in S} x_{ui} \bullet Y_i \bullet \gamma_u \tag{4}$$

where γ_u represents the duration of task execution.

(4) Compatibility Maximization($f_4(x)$)

Tasks are assigned to nodes that are compatible with their computational resource requirements:

$$f_4(x) = \sum_{u\in U} \sum_{i\in S} x_{ui} \bullet (1 - \delta(T_u^{type}, P_i)) \tag{5}$$

Where $\delta(T_u^{type}, P_i)$ is 1 if the task type matches the node resource type and 0 otherwise.

(5) Energy Minimization($f_5(x)$)

Energy efficiency is improved by minimizing the total energy consumption:

$$f_5(x) = \sum_{i\in S} \sum_{u\in U} x_{ui} \bullet E_i + \sum_{i,j\in S} x_{ui} \bullet x_{uj} \bullet D_{ij}^{node} \bullet E_{ij} \tag{6}$$

where E_{ij} denotes the energy cost of data transmission between nodes i and j

C. *Constraints*

In our computing network model, several constraints are considered to ensure the feasibility and efficiency of the optimization process:

(1) Task Assignment Constraint

Each task must be assigned to exactly one computational node:

$$\sum_{i\in S} x_{ui} = 1, \forall u \in U \tag{7}$$

(2) Node Capacity Constraint

The total workload assigned to a node cannot exceed its capacity:

$$\sum_{u\in U} x_{ui} \bullet T_u \le C_i, \forall i \in S \tag{8}$$

(3) Compatibility Constraint

Tasks can only be assigned to nodes with compatible resource types:

$$x_{ui} \bullet \delta(T_u^{type}, P_i) = 1, \forall u \in U, i \in S \tag{9}$$

(4) Bandwidth Constraint

The total data transmission for a node must not exceed its bandwidth:

$$\sum_{u \in U} x_{ui} \bullet \beta_u \le B_i, \forall i \in S \tag{10}$$

(5) Latency Tolerance Constraint

The communication latency for each task must be within its allowable threshold:

$$\sum_{i \in S} x_{ui} \bullet D_{iu}^{user} \le D_u^{\max}, \forall u \in U \tag{11}$$

IV. Multi-Objective Resource Optimization Based on Improved PSO for Computational Networks

In this section, we present the improved PSO algorithm, designed to optimize resource allocation and task scheduling in computational networks. The improved PSO algorithm enhances the classical PSO by integrating crossover and mutation mechanisms inspired by genetic algorithms (GA) and a dynamic parameter adjustment strategy. This hybrid approach combines the global exploration capabilities of PSO with the diversity maintenance of GA to achieve a better balance between exploration and exploitation, improving convergence speed and solution quality.

A. *Population Encoding and Initialization*

In the resource optimization problem, particle encoding is designed to represent the mapping of computational tasks to nodes in the network. Each particle's position is represented as a vector X_i, where **N** is the total number of tasks, and X_i^j denotes the node assigned to the j task. The initial positions of particles are randomly generated within the predefined bounds:

$$X_i^j \in \{1, 2, ..., M\} \tag{12}$$

Where M is the total number of computing nodes.

B. *Fitness Evaluation*

The fitness of each particle is evaluated using a predefined multi-objective fitness function F. This function incorporates multiple factors, including task completion latency, energy consumption, resource utilization, and cost. The function ensures that both individual task performance and overall network efficiency are optimized.

C. *Velocity and Position Update*

Particles update their velocities and positions iteratively to search for the optimal solution. The velocity update considers three components: the particle's personal best position (P , i), the global best position (G), and the current velocity:

$$\begin{gathered} V_i(t+1) = \omega \bullet V_i(t) + c_1 \bullet r_1 \bullet (P_i - X_i) \\ + c_2 \bullet r_2 \bullet (G - X_i) \end{gathered} \tag{13}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{14}$$

Where: ω is the inertia weight, dynamically adjusted to balance exploration and exploitation. c_1, c_2 are cognitive and social coefficients, respectively. r_1, r_2 are random numbers in [0, 1].

D. *Crossover and Mutation Mechanisms*

To maintain diversity and avoid premature convergence, the improved PSO incorporates crossover and mutation mechanis ms inspired by genetic algorithms:

Crossover: A portion of particles' positions are exchanged b etween two randomly selected particles to introduce new potent ial solutions:

$$X_i^j = \alpha \bullet X_i^j + (1-\alpha) \bullet X_k^j \tag{15}$$

Where X_k is the position of another randomly selected particle, and α is a crossover probability

Mutation: Particle positions are randomly perturbed with a small probability to explore unexplored regions of the search space:

$$X_i^j = X_i^j + \beta \bullet rand() \tag{16}$$

Where β is the mutation rate, and rand() generates a rando m number.

E. *Dynamic Parameter Adjustment*

To enhance convergence speed and maintain solution quality, the improved PSO dynamically adjusts the inertia weight (ω) and acceleration coefficients (c_1, c_2) over iterations:

$$\omega = \omega_{\max} - \frac{(\omega_{\max} - \omega_{\min})}{iter_{\max}} \bullet t \tag{17}$$

Where: $\omega_{\max}$ and $\omega_{\min}$ are the initial and final values of the inertia weight. t is the current iteration. $iter_{\max}$ is the maximum number of iterations.

F. *Termination Criteria*

The EEC-IHGWO algorithm iterates through the steps of position updating and hunting until the maximum number of iterations is met.

The improved PSO algorithm iterates through position updates, crossover, and mutation until one of the following conditions is met: The maximum number of iterations is reached. The improvement in fitness across iterations falls below a predefined threshold.

This enhanced PSO algorithm achieves efficient resource allocation and task scheduling in computational networks by leveraging the strengths of both PSO and GA.

V. Experiment and Results

In this section, the experimental setup is designed to evaluate the performance of the Improved Particle Swarm Optimization (IPSO) algorithm in solving the edge service deployment problem. The performance of IPSO is compared with the standard PSO and Genetic Algorithm (GA) across various metrics, including convergence speed, total delay, resource utilization, and deployment cost. The simulation parameters for these experiments are summarized in Table I.

TABLE I. SIMULATION PARAMETERS

Parameter	Value
Number of nodes	20
Number of tasks	50
Maximum iterations	50
Resource capacity per node	Random(50-150)
Communication delay(ms)	Random(1-10)
Deployment cost per node	Random(10-50)
Weight factor($\omega_1, \omega_2, \omega_3, \omega_4$)	0.4,0.3,0.2,0.1

The experiments were conducted in a simulated edge computing environment where each algorithm was applied to optimize task allocation across the available computational nodes. The goal was to minimize deployment cost and delay while maximizing resource utilization and maintaining compatibility with task requirements.

A. Convergence Analysis

In Fig. 1, the convergence performance of the proposed Improved PSO algorithm is compared with PSO and GA. The final values achieved by Improved PSO, PSO, and GA are approximately 4.1, 4.8, and 10.5, respectively. As shown in Fig. 1, the Improved PSO algorithm demonstrates faster convergence and better final optimization results compared to the other algorithms. The integration of crossover, mutation, and dynamic parameter adjustment in Improved PSO enhances its exploration and exploitation capabilities, allowing it to converge more effectively towards the optimal solution. Additionally, its hybrid nature ensures robust performance across various optimization problems, making it a more effective and scalable solution compared to traditional PSO and GA.

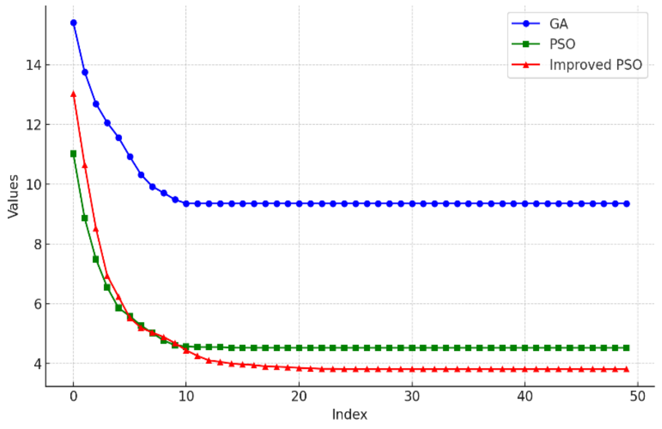

Fig. 1. Comparison of GA, PSO, and Improved PSO

B. Delay Optimization

In Fig. 2, the delay optimization performance of the proposed Improved PSO algorithm is evaluated against PSO and GA over 50 iterations. The final delay values obtained by Improved PSO, PSO, and GA are approximately 11, 14, and 40, respectively. The findings presented in Fig. 2 highlight the superior performance of the Improved PSO algorithm in minimizing network delay. The integration of crossover and mutation mechanisms, combined with dynamic parameter adjustment, significantly enhances the algorithm's capability to converge rapidly while maintaining a balance between exploration and exploitation. Furthermore, the Improved PSO exhibits robust optimization behavior, achieving consistently lower delay values across all iterations, underscoring its effectiveness and adaptability in complex optimization scenarios.

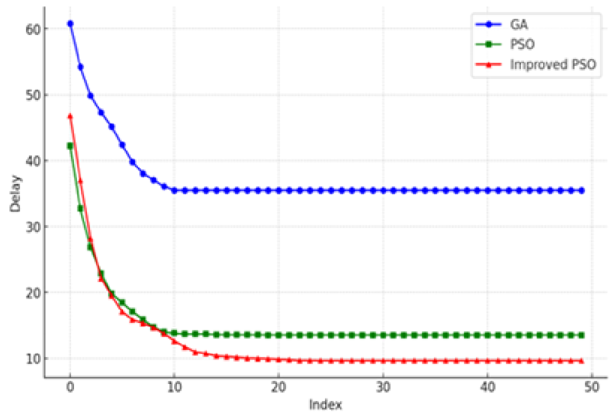

Fig 2 Delay Comparison of GA, PSO, and Improved PSO

C. Resource Utilization

The resource utilization results are shown in Fig. 3. IPSO achieves a utilization rate exceeding 80%, compared to 60% for PSO and 40% for GA. This indicates that IPSO effectively allocates resources across nodes, minimizing idle capacity and ensuring more balanced utilization. The dynamic parameter adjustment in IPSO contributes to its ability to allocate tasks to nodes with the highest availability while avoiding overloading any single node.

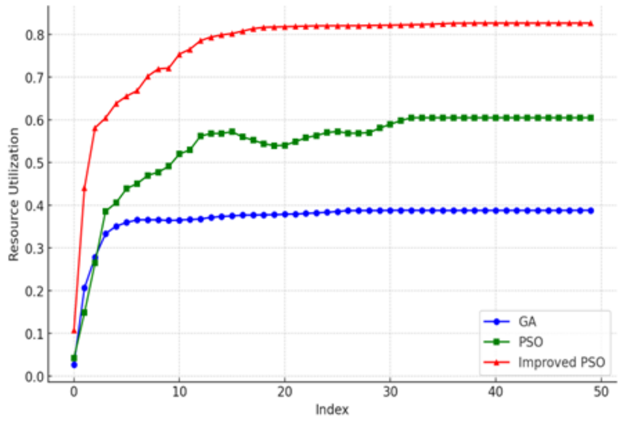

Fig 3 Resource Utilization Comparison of GA, PSO, and Improved PSO

D. Deployment Cost

The deployment cost comparison is presented in Fig. 4. IPSO achieves the lowest deployment cost of approximately 1.0 unit, compared to 2.0 units for PSO and 2.5 units for GA. This demonstrates IPSO's capability to find cost-effective solutions while considering the constraints of computational resources and task requirements. The reduction in deployment cost reflects IPSO's efficiency in balancing performance objectives and resource expenses.

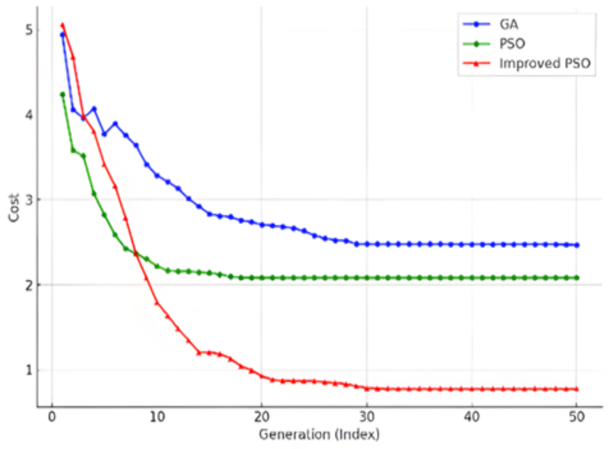

Fig 4 Cost Optimization Comparison of GA, PSO, and Improved PSO

E. *Network Scalability*

To evaluate the scalability of the algorithms, experiments were conducted under varying numbers of computational nodes. The results, summarized in Table II, provide a comprehensive comparison of the algorithms' performance in terms of total delay, resource utilization, and deployment cost. IPSO consistently outperforms PSO and GA across all configurations, demonstrating its robustness and adaptability in larger-scale networks.

Table II. Algorithm Performance Across Different Node Configurations

Nodes	Metric	IPSO	PSO	GA
20	Total Delay	10.0	15.0	20.0
	Utilization	80.0	70.0	60.0
	Deployment Cost	1.0	2.0	3.0
30	Total Delay	9.5	14.5	19.5
	Utilization	82.0	72.0	61.0
	Deployment Cost	1.1	2.2	3.1
40	Total Delay	9.0	13.0	18.0
	Utilization	84.0	73.0	62.0
	Deployment Cost	1.2	2.4	3.2
50	Total Delay	8.5	12.5	17.5
	Utilization	86.0	75.0	64.0
	Deployment Cost	1.3	2.6	3.4

The total delay results across varying node configurations are depicted in Figure 4. As shown in the table and the figure, IPSO achieves the lowest total delay across all configurations. For instance, at 20 nodes, IPSO has a total delay of 10.0 ms, significantly lower than 15.0 ms for PSO and 20.0 ms for GA. As the network scales to 50 nodes, the delay for IPSO reduces further to 8.5 ms, whereas PSO and GA exhibit delays of 12.5 ms and 17.5 ms, respectively. These results highlight the effectiveness of IPSO in minimizing communication overhead, which is critical for improving the overall performance in larger networks.

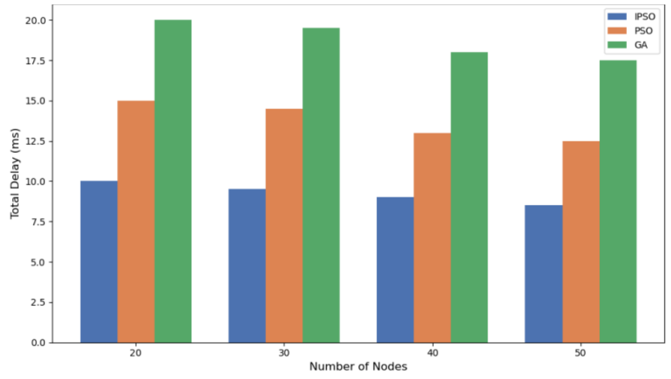

Fig 5 . Total delay comparison across node configurations.

The resource utilization results are presented in Figure 5, which illustrates the efficiency of IPSO in maximizing resource usage. The utilization for IPSO improves with the network size, reaching 86.0% at 50 nodes, compared to 75.0% for PSO and 64.0% for GA. This significant improvement demonstrates IPSO's ability to allocate resources more effectively, ensuring higher utilization rates even in large-scale networks. The enhanced resource utilization also reflects the optimization mechanisms of IPSO, which balance workload distribution and minimize resource wastage.

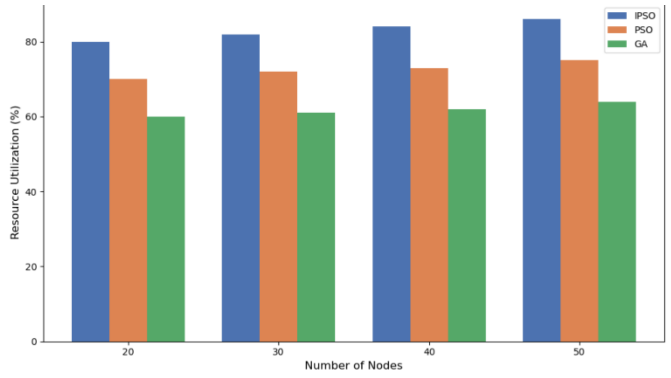

Fig 6. Resource utilization comparison across node configurations.

Deployment cost is another critical factor in evaluating the scalability of algorithms, as shown in Figure 6. IPSO consistently maintains the lowest deployment cost across all configurations. For example, at 20 nodes, IPSO incurs a cost of 1.0 units, significantly lower than 2.0 units for PSO and 3.0 units for GA. At 50 nodes, IPSO still outperforms with a cost of 1.3 units, compared to 2.6 units for PSO and 3.4 units for GA. These results demonstrate IPSO's capability to optimize task deployment while maintaining low costs, making it a more efficient solution for resource-constrained networks.

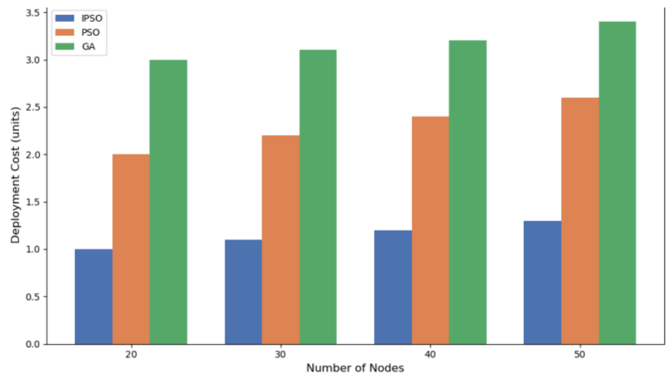

Figure 7. Deployment cost comparison across node configurations.

The analysis of the results in Table II and Figures 5–7 demonstrates the superior scalability of IPSO compared to PSO and GA. The consistent improvements in total delay, resource utilization, and deployment cost highlight IPSO's ability to handle larger network scales effectively. These results validate the robustness and adaptability of IPSO in optimizing computational resource allocation in edge computing scenarios. The findings further suggest that IPSO is well-suited for dynamic and large-scale network environments, making it a reliable choice for computational service deployment in resource-constrained scenarios

VI. Conclusion

In this paper, a multi-objective optimization model for edge computing service deployment in computational networks is proposed, addressing key metrics such as computational resource types, node-to-node latency, node-to-user latency, and deployment cost. An improved Particle Swarm Optimization (IPSO) algorithm is introduced, incorporating crossover, mutation, and dynamic parameter adjustment to optimize task allocation and resource utilization. Comparative experiments demonstrate that IPSO significantly outperforms traditional PSO and GA in terms of convergence speed, delay minimization, resource utilization, and cost reduction. The results indicate the proposed model's effectiveness in enhancing system performance while reducing deployment costs, especially in large-scale computational network scenarios.

In future work, we plan to investigate the integration of reinforcement learning and advanced metaheuristic algorithms to further improve the adaptability and scalability of task scheduling in dynamic and heterogeneous edge computing environments.

References

[1] B. Ali, M. Adeel Pasha, S. u. Islam, H. Song and R. Buyya, "A Volunteer-Supported Fog Computing Environment for Delay-Sensitive IoT Applications," in *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3822-3830, 1 March1, 2021, doi: 10.1109/JIOT.2020.3024823.

[2] J. Liu, G. Shou, Y. Liu, Y. Hu and Z. Guo, "Performance Evaluation of Integrated Multi-Access Edge Computing and Fiber-Wireless Access Networks," in *IEEE Access*, vol. 6, pp. 30269-30279, 2018, doi: 10.1109/ACCESS.2018.2833619.

[3] H. Song, B. Gu, K. Son and W. Choi, "Joint Optimization of Edge Computing Server Deployment and User Offloading Associations in Wireless Edge Network via a Genetic Algorithm," in *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2535-2548, 1 July-Aug. 2022, doi: 10.1109/TNSE.2022.3165372.

[4] L. Chen, S. Zhou and J. Xu, "Computation Peer Offloading for Energy-Constrained Mobile Edge Computing in Small-Cell Networks," in *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619-1632, Aug. 2018, doi: 10.1109/TNET.2018.2841758.

[5] S. -E. Chafi, Y. Balboul, M. Fattah, S. Mazer and M. El Bekkali, "Novel PSO-Based Algorithm for Workflow Time and Energy Optimization in a Heterogeneous Fog Computing Environment," in *IEEE Access*, vol. 12, pp. 41517-41530, 2024, doi: 10.1109/ACCESS.2024.3377236.

[6] V. Abedifar, M. Eshghi, S. Mirjalili and S. M. Mirjalili, "An optimized virtual network mapping using PSO in cloud computing," *2013 21st Iranian Conference on Electrical Engineering (ICEE)*, Mashhad, Iran, 2013, pp. 1-6, doi: 10.1109/IranianCEE.2013.6599723.

[7] R. Vadivel and S. Muthu T., "An effective HPSO-MGA Optimization Algorithm for Demand based Resource Allocation in Cloud Environment," *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2020, pp. 1189-1194, doi: 10.1109/ICACCS48705.2020.9074442.

[8] Y. Alhaizaey, J. Singer and A. L. Michala, "Optimizing Heterogeneous Task Allocation for Edge Compute Micro Clusters Using PSO Metaheuristic," *2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC)*, Paris, France, 2022, pp. 1-8, doi: 10.1109/FMEC57183.2022.10062755.

[9] S. Singh and D. H. Kim, "Joint Optimization of Computation Offloading and Resource Allocation in C-RAN With Mobile Edge Computing Using Evolutionary Algorithms," in *IEEE Access*, vol. 11, pp. 112693-112705, 2023, doi: 10.1109/ACCESS.2023.3322650.

[10] K. Wang, K. Yang and C. S. Magurawalage, "Joint Energy Minimization and Resource Allocation in C-RAN with Mobile Cloud," in *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760-770, 1 July-Sept. 2018, doi: 10.1109/TCC.2016.2522439.

[11] S. Midya, A. Roy, K. Majumder and S. Phadikar, "PSO based Optimized Resource Allocation in three tier Cloud Architecture for VANET," *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, Kolkata, India, 2018, pp. 12-17, doi: 10.1109/ICRCICN.2018.8718723.