International Journal of Computer Vision and Signal Processing, 15(1), 112-119(2025) ORIGINAL ARTICLE

Optimizing Task Offloading in Fog Computing with HAGSA-NS: A Hybrid Adaptive Gravitational Search Algorithm

Md. Emran Biswas[2], Tangina Sultana[2], Md. Delowar Hossain[1], Mst. Khadeja Sarker[2], Ga-Won Lee[3], Eui-Nam Huh[3]

[1] *Department of Computer Science and Engineering, Hajee Mohammad Danesh Science & Technology University, Dinajpur 5200, Bangladesh.*

[2] *Department of Electronics and Communication Engineering, Hajee Mohammad Danesh Science & Technology University, Dinajpur 5200, Bangladesh.*

[3] *Department of Computer Science and Engineering, Kyung Hee University, Yongin-si 17104, Korea*

[*] *Correspondence: johnhuh@khu.ac.kr*

Abstract

The swift proliferation of Internet of Things (IoT) devices and the demand for minimal latency applications has rapidly increased the incorporation of fog computing alongside traditional cloud computing. Nevertheless, efficient task offloading in fog settings remains a significant issue due to variable network conditions, resource constraints, and stringent quality-of-service (QoS) requirements. This research proposes a novel Hybrid Adaptive Gravitational Search Algorithm with Neighborhood Search (HAGSA-NS) to enhance task offloading in fog computing, solving these issues. HAGSA-NS integrates the global exploration capabilities of the Adaptive Gravitational Search Algorithm (AGSA), the diversity-enhancing features of Differential Evolution (DE), and the local exploitation benefits of Neighborhood Search (NS). This hybrid approach enables effective and efficient task allocation, even in highly dynamic and resource-constrained environments. The effectiveness of HAGSA-NS is evaluated against two prevalent optimization methods, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), using two primary metrics: ideal fitness values across iterations and average computational latency. Experimental results demonstrate that HAGSA-NS consistently outperforms PSO and GA, achieving reduced fitness values and significantly decreased delays. HAGSA-NS achieves an average delay of 1.0, whereas PSO exhibits a delay of 4.234 and GA demonstrates a delay of 1.791. These findings highlight the benefits of HAGSA-NS in terms of solution quality and computational efficiency, positioning it as a viable strategy for work offloading in fog computing environments.

Keywords: Fog Computing, Task Offloading, HAGSA-NS, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Internet of Things (IoT), Quality of Service (QoS), Computational Efficiency, Delay Minimization, Resource Allocation

IJCVSP

ISSN: 2186-1390 (Online)
http://cennser.org/IJCVSP

Article History:
Received: 10/4/2025
Revised: 13/7/2025
Accepted: 1/11/2025
Published Online: 23/11/2025

Email addresses: emran.hstu1999@gmail.com (Md. Emran Biswas[2]), tangina@hstu.ac.bd (Tangina Sultana[2]), delowar@khu.ac.kr (Md. Delowar Hossain[1]), bristykhadeja@gmail.com (Mst. Khadeja Sarker[2]), gawon@khu.ac.kr (Ga-Won Lee[3]), johnhuh@khu.ac.kr (Eui-Nam Huh[3])

1. INTRODUCTION

The instantaneous proliferation of Internet of Things (IoT) devices andThe growing requirement for minimal latency,outstanding performance applications have necessitated the evolution of traditional cloud computing architectures [1]. Fog computing has emerged as a feasible option, situating processing resources nearer to The edge of

the network, thereby decreasing latency and bandwidth use [2]. The efficient allocation and offloading of tasks in fog computing settings present difficulties because to the network's fluctuating nature, limited resources, and rigorous quality-of-service (QoS) requirements [3].

Task offloading, the transfer of computational responsibilities from end-user devices to fog nodes, is an essential aspect of fog computing. The primary aim is to minimize delays, optimize load distribution, and ensure reliable task execution while adhering to constraints such as deadlines and resource availability. It optimizes resource usage by transferring computationally intensive tasks from resource-constrained devices to more powerful cloud or fog nodes, which improves overall system performance. Offloading reduces latency by enabling faster processing at the edge in fog computing, minimizing the distance data needs to travel. Additionally, it enhances energy efficiency by conserving battery life on local devices, which is crucial for mobile and IoT applications [4]. Cloud computing further supports scalability, allowing the system to dynamically handle varying workloads and large-scale applications. Overall, task offloading in these paradigms results in cost savings by reducing the need for local infrastructure and leveraging the capabilities of external resources [5].

Classical optimization approaches, such as Genetic Algorithms (GA) [6] [7] and Particle Swarm Optimization (PSO) [8] [9], have been extensively employed in the context of job offloading, owing to their ability to search for optimal or near-optimal solutions in large, complex search spaces. These methods have demonstrated considerable success in many applications, leveraging their stochastic nature to explore diverse solutions and converge to effective results. However, despite their advantages, these techniques often encounter challenges when it comes to balancing exploration with the refinement of existing solutions in dynamic, ever-changing environments. Specifically, the inherent trade-off between searching for new, potentially better solutions and optimizing already identified ones can lead to suboptimal outcomes, particularly in scenarios where the problem landscape is highly variable or uncertain. As a result, these classical methods may struggle to maintain consistent optimal performance, especially when applied to real-world, complex job offloading scenarios where system conditions and requirements are continuously evolving.

We present HAGSA-NS, a Hybrid Adaptive Gravitational Search Algorithm with Neighborhood Search, to address these challenges in efficient work offloading inside fog computing. HAGSA-NS amalgamates the global exploration proficiency of the Adaptive Gravitational Search Algorithm (AGSA) with the local exploitation efficacy of Neighborhood Search (NS) and the diversity-enhancing characteristics of Differential Evolution (DE). This hybrid model ensures effective and efficient task allocation, particularly in highly dynamic and resource-constrained fog circumstances.

The major contributions of this work are as follows:

- We offer **HAGSA-NS**, a new hybrid optimization approach that amalgamates AGSA, DE, and NS for the enhancement of work offloading in fog computing.
- We propose a dynamic adaptation technique in AGSA to improve convergence and avoid local optima.
- We incorporate Neighborhood Search to enhance local exploitation, enabling more rapid and accurate job allocation.
- We assess the efficacy of HAGSA-NS using comprehensive computations, demonstrating its superiority over prominent methods such as GA and PSO in terms of delay reduction, load allocation, and reliability.

This document is structured as follows: Section II examines pertinent literary works. Section III comprehensively explains the process. Section IV presents the results, while Section V closes the study.

2. RELATED WORKS

Task offloading in fog computing has garnered substantial attention in recent years because of its ability to alleviate the limitations of traditional cloud computing. This section examines the existing research on task offloading, optimization techniques, and their applicability inside fog computing systems.

2.1. Task Offloading in Fog Computing

The task offloading challenge in fog and edge computing environments has been extensively examined in recent years, with many studies concentrating on different optimization methodologies, issues, and solutions. Alasmari et al. (2023) [10] focus on improving energy accuracy and network efficiency in fog computing by employing multi-classifiers for job offloading in resource-constrained environments. Their findings underscores the importance of enhancing job offloading techniques to conserve energy and optimize network performance in fog situations. Goel et al. (2023) [11] present a comprehensive examination of work offloading and load balancing strategies in fog computing, clarifying key optimization techniques and emphasizing recent advancements in the field. Their investigation evaluates various load balancing approaches and continual demands for improvement in fog systems. Wei et al. (2023) [12] introduce a trading-based framework employing Multi-Agent Generalized Advantage Estimation (MA-GAC) for task offloading in fog computing for vehicles. Their technology, focused on deep reinforcement learning, seeks to improve system welfare and maximise resource allocation, particularly inside vehicle networks. Wu et al. (2023) [13] provide a delay-sensitive job offloading technique employing Semi-Markov Decision Processes (SMDP) for Vehicle-Fog Computing (VFC)-assisted platoons. Their research illustrates how this methodology reduces offloading delays,

especially in vehicle platoons, yielding encouraging outcomes from simulation-based validation. Tran-Dang and Kim (2023) [14]study fog computing systems' dynamic collaborative task offloading, emphasizing the reduction of task delays through the use of parallel processing across various fog devices. Their methodology seeks to minimize job processing duration while preserving low computational complexity. Premalatha and Prakasam (2024) [15] provide the technique for IoT-Fog networks called Optimal Energy-efficient Resource Allocation (OEeRA), which improves energy use and job offloading efficiency through Fault Isolation Recovery (FIR) and Multi-Class Resource Allocation (MCRA), resulting in significant performance enhancements. Rezaee et al. (2024) [16] examine workload offloading and management strategies in IoT-Fog-Cloud ecosystems. Their article emphasizes the significance of fog computing in handling the substantial data flood from IoT devices, mitigating network congestion, and enhancing resource allocation in fog nodes. Jain and Kumar (2023) [17] provide a Deep Reinforcement Learning (DRL) methodology for job offloading in IoT-Fog-Cloud architectures. Their methodology attains a 96.23% task deadline fulfillment rate and an 8.25% enhancement in performance, demonstrating the efficacy of DRL in optimizing job offloading in fog situations. Mahapatra et al. (2024) [18] provide a six-layer architecture that incorporates cloud, fog, edge, mist, dew, osmotic, and hybrid computing. They investigate the interaction of these paradigms to enhance work scheduling, load balancing, and resource allocation, providing significant insights for future developments in task offloading methodologies. P4-assisted task offloading is proposed by Akyildiz et al. (2023) [19] for fog-based IoT networks. The TOS-P4 strategy decreases waiting times by a factor of 6.54 relative to conventional approaches, demonstrating the efficacy of P4 technology in enhancing task processing efficiency. Dash et al. (2023) [20] research a Quality of Service (QoS) aware task offloading strategy using Software-Defined Networking (SDN) and the Golden Jackal optimization algorithm for useful task scheduling in fog computing situations. Their approach improves performance for latency-sensitive applications, guaranteeing optimum task offloading under diverse network circumstances. Kar et al. (2023) [21] investigate offloading techniques inside federated cloud-edge-fog systems, highlighting the application of reinforcement learning to enhance task offloading decisions. Their investigation offers significant insights into conventional optimization and machine learning methodologies for IoT traffic offloading inside these federated systems. Akhlaqi and Hanapi (2023) [22] present a thorough study on task offloading in Mobile Edge Computing (MEC), analyzing the problems, optimization strategies, and algorithms employed in various domains such as IoT, autonomous cars, and 5G. Their analysis indicates potential avenues for further investigation to tackle the emerging issues in MEC task offloading. These papers jointly enhance comprehension of task offloading in fog, edge, and mobile computing settings, emphasizing critical optimization methodologies, problems, and upcoming technologies in this field.

2.2. Optimization Algorithms for Task Offloading

In order to solve the issues of work scheduling, energy consumption, and delay minimisation in cloud-fog computing systems, recent research has presented creative methods. Khiat et al. (2024) [23] presented the GAMMR algorithm, which optimizes energy usage and reaction time, surpassing conventional genetic algorithms. Saif et al. (2023) [24] introduced the Multi-Objectives Grey Wolf Optimizer (MGWO) to enhance task allocation, hence minimizing delay and energy expenditure. Liu et al. (2023) [6] introduced a genetic algorithm for efficient computation offloading, enhancing resource usage. Matrouk et al. (2023) [25] devised the MISSION technique to optimize task scheduling and resource allocation, hence improving performance metrics like as energy usage and latency. With an emphasis on energy efficiency, Agarwal et al. (2023) [7] developed a hybrid genetic technique for multiprocessor task planning. In support of load balancing and quality of experience (QoE), Baburao et al. (2023) [8] developed a PSO-based increased dynamic resource allocation technique. Ogundoyin et al. (2023) [26] integrated Particle Swarm Optimization and Firefly algorithms for the optimal selection of fog nodes, enhancing resource usage and energy efficiency. Vispute and Vashisht (2023) [9] presented the EETSPSO algorithm for energy-efficient task scheduling, surpassing current methodologies. Saif et al. (2023) [27] presented the NPSO method for workload allocation, effectively decreasing energy consumption and latency while lowering the maximum delay threshold. These studies illustrate the efficacy of metaheuristic algorithms in optimizing distributed computing systems, yielding enhanced performance regarding energy consumption, latency, and resource use.

2.3. Research Gap and Contribution

Considering advancements in work offloading and optimization, many problems remain unsolved. Most techniques primarily focus on either global exploration or local exploitation, but seldom both concurrently. Secondly, the dynamic and varied attributes of fog computing environments require algorithms that can adjust to changing conditions. Ultimately, there is a lack of hybrid optimization approaches specifically designed for task offloading in fog computing. To address these shortcomings, we provide **HAGSA-NS**, a hybrid algorithm that amalgamates the global exploration capabilities of AGSA, the diversity-enhancing features of DE, and the local exploitation strengths of NS. Our technique is specifically designed for efficient task offloading in fog computing, ensuring robust performance in dynamic and resource-constrained environments.

3. Methodology

The proposed strategy for effective job offloading in fog computing utilizes the Hybrid Adaptive Gravitational Search Algorithm with Neighborhood Search (HAGSA-NS). This section describes the workflow of HAGSA-NS, as illustrated in Figure 1, and explains each step in detail.

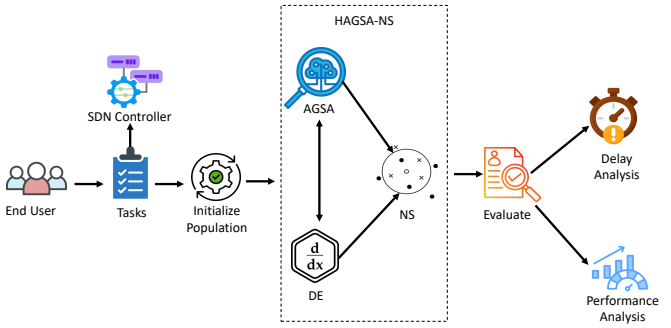

Figure 1: Workflow of the proposed HAGSA-NS model for task offloading in fog computing.

3.1. Data Collection

The initial phase of the suggested technique entails data collection, which encompasses the acquisition of information on tasks, fog nodes, and network circumstances. Tasks are produced by end-user devices (EUs) and defined by properties like size, deadline, and processing needs. Fog nodes (FNs) are defined by their processing power, failure frequency, and geographic position. The SDN controller functions as the primary manager, aggregating real-time information on the network's condition, including latency, bandwidth, and resource availability. This data functions as the input for the HAGSA-NS algorithm, facilitating informed judgments on task offloading.

3.2. Proposed Optimization Algorithm

The essence of the suggested technique is the hybrid optimization process, which integrates the advantages of the Adaptive Gravitational Search Algorithm (AGSA), Differential Evolution (DE), and Neighborhood Search (NS). The HAGSA-NS method proceeds with the initialization of a population of solution candidates, with every possible outcome denoting a prospective task assignment to fog nodes. The placements and velocities of the particles in the population are produced randomly within specified limits. The efficacy of each solution is assessed according to the cumulative delay, encompassing queuing delay, offloading delay, and transmission delay. AGSA is utilized for global exploration, wherein particles (solutions) traverse the search space influenced by gravity forces. The gravitational constant G is iteratively adjusted to equilibrate exploration and exploitation. The particles' locations and velocities are modified With regard to these equations:

$$\psi_i(\tau+1) = w \cdot \psi_i(\tau) + \sum_{j \neq i} \frac{G \cdot N_j}{R_{ij}^2} \cdot (\xi_j(\tau) - \xi_i(\tau)), \quad (1)$$

$$\xi_i(\tau+1) = \xi_i(\tau) + \psi_i(\tau+1), \quad (2)$$

where $\psi_i(\tau)$ and $\xi_i(\tau)$ represent the velocity and position of particle i at iteration τ, w is the inertia weight, N_j is the mass of particle j, and R_{ij} is the distance between particles i and j.

DE was integrated into HAGSA-NS to improve variation and prevent early convergence. DE uses its mutation and crossover procedures on the population to provide fresh candidate solutions. Definition of the mutation operation:

$$\phi_i = \phi_{r1} + T \cdot (\phi_{r2} - \phi_{r3}), \quad (3)$$

where ϕ_{r1}, ϕ_{r2}, and ϕ_{r3} are randomly selected particles, and T is the mutation parameter. The crossover functionality combines the mutated solution with the original solution to produce a trial solution.

Local exploitation using NS helps to refine the solutions in a limited search area. By means of NS, which investigates the neighborhood of every solution to identify better assignments, the finest solutions from AGSA and DE are further enhanced. This stage guarantees the convergence of the method toward excellent answers. Algorithm presents the pseudo-code for the HAGSA-NS algorithm. 1.

The HAGSA-NS algorithm commences by initializing the population of N particles, where every particle x_i denotes a prospective task assignment to fog nodes. The positions $X = \{x_1, x_2, \ldots, x_N\}$ and velocities $V = \{v_1, v_2, \ldots, v_N\}$ of the particles are randomly generated within predefined bounds. The gravitational constant G controls the strength of gravitational forces, while the weight of inertia w balances the effect of the particle's present velocity. The mutation factor F is used in Differential Evolution (DE) to generate new candidate solutions. The fitness $f(x_i)$ of each particle is evaluated based on the total delay, which includes queuing delay, offloading delay, and transmission delay. Individual best p_i and general best g solutions are updated iteratively.. The gravitational forces F_i are calculated using the masses M_j of particles and the distances R_{ij} between them. The positions and velocities of the particles are updated using the gravitational forces, and DE is applied to enhance diversity. Neighborhood Search (NS) is performed on elite solutions X_{elite} to refine the solutions locally. The gravitational constant G is adaptively updated over iterations to balance exploration and exploitation. After all, the best solution $X^* = g$ is returned as the optimal task assignment.

3.3. Baseline Optimization Algorithms

The efficacy of the proposed HAGSA-NS algorithm is evaluated using two commonly applied baseline optimization methods: GA and PSO. The following are descriptions of these algorithms:

Algorithm 1 Hybrid Adaptive Gravitational Search Algorithm with Neighborhood Search (HAGSA-NS)

Require: Population size N, number of tasks T, maximum iterations I_{max}, elite size E
Ensure: Optimal task assignments X^*

Initialize population $X = \{x_1, x_2, \ldots, x_N\}$ and velocities $V = \{v_1, v_2, \ldots, v_N\}$
Initialize gravitational constant G, inertia weight w, mutation factor F
for $t = 1$ to I_{max} **do**
  Evaluate fitness $f(x_i)$ for each particle $x_i \in X$
  Update individual best p_i and general best g
  Calculate gravitational forces F_i for each particle x_i:

$$F_i = \sum_{j \neq i} \frac{G \cdot M_j}{R_{ij}^2} \cdot (x_j - x_i)$$

  Update velocities v_i and positioning x_i for each particle:

$$v_i(t+1) = w \cdot v_i(t) + F_i$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

  Perform mutation and crossover using DE:
  for $i = 1$ to N **do**
    Select random particles x_{r1}, x_{r2}, x_{r3}
    Generate mutant $v_i = x_{r1} + F \cdot (x_{r2} - x_{r3})$
    Perform crossover to generate trial solution u_i
    if $f(u_i) < f(x_i)$ **then**
      Replace x_i with u_i
    end if
  end for
  Perform neighborhood search (NS) on elite solutions X_{elite}:
  for $x_i \in X_{elite}$ **do**
    Generate trial solution $x_i' = x_i + \Delta x$, where Δx is a small random step
    if $f(x_i') < f(x_i)$ **then**
      Replace x_i with x_i'
    end if
  end for
  Update gravitational constant G:

$$G = G \cdot \exp(-\alpha \cdot t / I_{max})$$

  Integrate global best solution g into population X
end for
return Global best solution $X^* = g$

3.3.1. Genetic Algorithm (GA)

GA [28] is a population-based optimization technique inspired by the process of natural selection. It evolves a population of candidate solutions over multiple generations using genetic operators such as selection, crossover, and mutation The crossover operation is defined as:

$$x_{child} = \alpha \cdot x_{parent1} + (1 - \alpha) \cdot x_{parent2} \tag{4}$$

where α is a random number between 0 and 1. After crossover, mutation is applied to the offspring to introduce diversity. The mutation operation is defined as:

$$x_{mutated} = x_{child} + \Delta x \tag{5}$$

where Δx is a small random perturbation. The GA repetitively executes these stages till a halting threshold, such as the highest number of generations, is satisfied. The most effective option identified throughout the optimization process is presented as the ideal alternative.

3.3.2. Particle Swarm Optimization (PSO)

PSO [29] is a population-centric optimization process derived from the social behaviors shown by birds in flocks or fish in groups. It employs a swarm of particles, with each particle symbolizing a proposed solution. The particles traverse the search space according to their velocity, which is affected by each one's optimal location and the swarm's global suitable value.

The velocity of each particle is updated using:

$$\phi_i(\tau+1) = w \cdot \phi_i(\tau) + \kappa_1 \cdot \rho_1 \cdot (\psi_i - \xi_i(\tau)) + \kappa_2 \cdot \rho_2 \cdot (\gamma - \xi_i(\tau)) \tag{6}$$

where $\phi_i(\tau)$ represents the velocity of particle i at iteration τ, w is the inertia weight, κ_1 and κ_2 are the cognitive and social acceleration coefficients, ρ_1 and ρ_2 are random values between 0 and 1, ψ_i is the personal best position of particle i, and γ is the global best position of the swarm. The position of each particle is updated using:

$$\xi_i(\tau + 1) = \xi_i(\tau) + \phi_i(\tau + 1) \tag{7}$$

The PSO method incrementally modifies the velocities and placements of the particles until a termination requirement, such as a maximum iteration count, is satisfied. The global optimal outcome g is provided as the best solution.

3.4. Simulation Setup

This study evaluates the performance of three metaheuristic algorithms—Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Hybrid Adaptive Gained Simulated Annealing with Neighborhood Search (HAGSA-NS)—for solving the task scheduling problem in a distributed computing environment. Each algorithm is simulated using a uniform experimental setup consisting of a population size of 110 and a maximum of 600 iterations. The evaluation is carried out based on three key performance metrics: Best Fitness, Average Delay, and Resource Utilization Rate. The dataset comprises dependent tasks with associated execution times, deadlines, and a fixed number of resources.

3.4.1. Best Fitness

The Best Fitness metric (f_{best}) evaluates the quality of the optimal solution obtained by an algorithm during the simulation. It corresponds to the minimum fitness value observed across all generations and reflects how well the algorithm minimizes the objective function.

$$f_{\text{best}} = \min_{i \in \{1,2,\ldots,I\}} f(S_i) \tag{8}$$

Here, I is the total number of iterations, $f(S_i)$ represents the fitness value of the best individual at iteration i, and S_i denotes the solution (i.e., task schedule) at that iteration. A lower f_{best} implies a better-performing algorithm. This metric is visualized using convergence curves over iterations to illustrate the search behavior and convergence stability.

3.4.2. Average Delay

The Average Delay metric ($\bar{D}$) quantifies the average delay for tasks that miss their deadlines. It is an important indicator of schedule quality in time-sensitive applications. Only tasks with delays ($F_i > D_i$) are considered.

$$\bar{D} = \frac{1}{|\mathcal{L}|} \sum_{\tau_i \in \mathcal{L}} (F_i - D_i), \quad \text{where} \quad \mathcal{L} = \{\tau_i \mid F_i > D_i\} \tag{9}$$

In this equation, F_i is the finish time of task τ_i, D_i is its deadline, and $\mathcal{L}$ is the set of delayed tasks. The metric penalizes schedules with many delayed tasks or large delays, providing insight into the temporal efficiency of the scheduling strategy.

All metrics were calculated after the final iteration using the best solution found by each algorithm. This setup ensures a fair and consistent comparison across GA, PSO, and HAGSA-NS under identical experimental conditions.

4. Results

In this section, we present a comprehensive analysis of the performance and delay characteristics of the proposed algorithm in comparison with PSO and GA. The evaluation is conducted based on two key metrics: the best fitness values achieved over iterations and the average delay incurred by each algorithm. Comparative graphs are applied to represent the results, offering insights into the algorithms' effectiveness and efficacy.

4.1. Performance Comparison

The performance of the algorithms is evaluated by analyzing the convergence behavior and the quality of the solutions obtained over iterations. Figure 2 illustrates the best fitness values achieved by HAGSA-NS, PSO, and GA across iterations.

The HAGSA-NS algorithm that has been proposed exhibits superior performance, consistently attaining lower fitness values than GA and PSO. This suggests that it is capable of effectively balancing exploration and exploitation, which allows it to converge to superior solutions and avoid local optima. Although PSO is competitive, it exhibits a propensity to reach a plateau at an earlier stage, which implies that convergence may occur prematurely in specific circumstances. Conversely, GA demonstrates delayed convergence and higher fitness values, which underscore its limitations in managing intricate optimization landscapes. The findings emphasize the efficacy of HAGSA-NS in attaining higher-quality solutions and faster convergence.

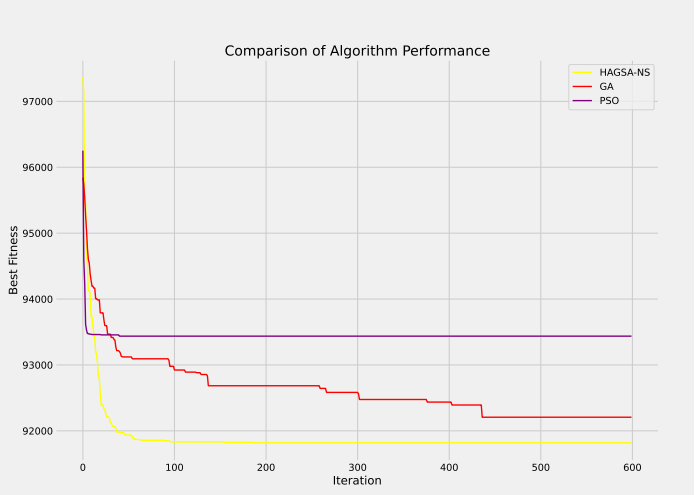

Figure 2: Comparison of Algorithm Performance: HAGSA-NS, PSO, and GA. The plot shows the best fitness values over iterations for each algorithm.

4.2. Delay Comparison

In addition to performance, the average delay incurred by each algorithm is analyzed to evaluate their computational efficiency. The average delay values for HAGSA-NS, PSO, and GA are 1.0, 4.234, and 1.791, respectively. Figure 3 provides a visual representation of these delay values across the algorithms.

The computational efficiency of HAGSA-NS is underscored by its ability to attain the lowest average latency. This is due to the adaptive mechanisms and neighborhood search strategy, which enhance convergence speed and minimize superfluous computations. The computational overhead of PSO is comparatively higher, as evidenced by the maximum delay. Although GA is superior to PSO, it still results in a longer delay than HAGSA-NS.

5. Conclusion

The HAGSA-NS, a novel optimization algorithm, was introduced in this paper to resolve the challenges of task outsourcing in fog computing environments. HAGSA-NS efficiently converges to high-quality solutions by successfully integrating the global exploration capabilities of AGSA,

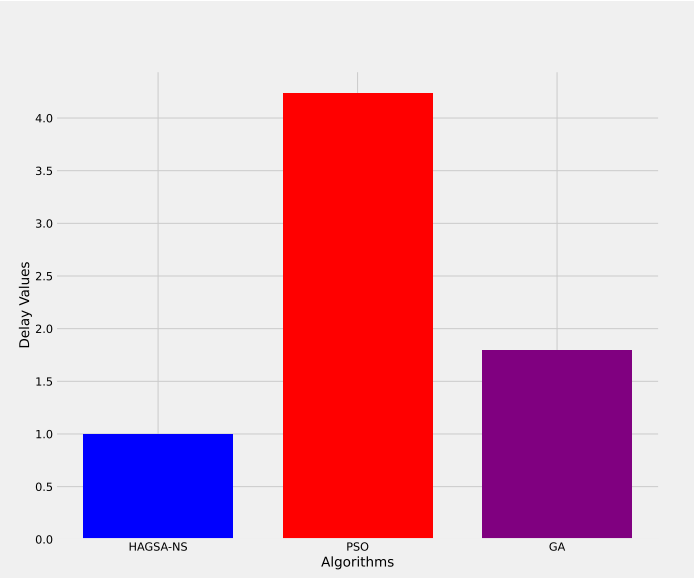

Figure 3: Comparison of Average Delay Across Algorithms: HAGSA-NS, PSO, and GA. The plot shows the delay values for each algorithm.

the diversity-enhancing properties of DE, and the local exploitation strengths of NS. A better balance between exploration and exploitation is attained in order to achieve this. The proposed algorithm was assessed against two state-of-the-art optimization techniques, PSO and GA, based on two critical metrics: the average computational latency and the highest fitness values over iterations. The findings indicated that HAGSA-NS consistently outperforms PSO and GA, resulting in significantly reduced delays and decreased fitness values. In particular, HAGSA-NS attained an average latency of 1.0, which is lower than the 4.234 for PSO and the 1.791 for GA. These results emphasize the efficacy of HAGSA-NS in mitigating delays, balancing burdens, and guaranteeing the reliable execution of tasks in dynamic and resource-constrained fog environments. HAGSA-NS is a robust and efficient solution for task outsourcing due to its neighborhood search strategy and adaptive mechanisms, which allow it to avoid local optima and achieve quicker convergence.

The future work will concentrate on the extension of HAGSA-NS's applicability to more complex and real-world fog computing scenarios, as well as the exploration of its integration with additional improving strategies to enhance performance even more. The findings of this investigation underscore the potential of HAGSA-NS as a potent instrument for optimizing task offloading in fog computing, thereby facilitating the development of high-performance, low-latency IoT applications.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2023-00220631, Edge Cloud Reference Architecture Standardization for Low Latency and Lightweight Cloud Service). Eui-Nam Huh and Md. Delowar Hossain are the co-corresponding author.

References

[1] N. A. Angel, D. Ravindran, P. D. R. Vincent, K. Srinivasan, Y.-C. Hu, Recent advances in evolving computing paradigms: Cloud, edge, and fog technologies, Sensors 22 (1) (2021) 196.
[2] A. Hazra, P. Rana, M. Adhikari, T. Amgoth, Fog computing for next-generation internet of things: fundamental, state-of-the-art and research challenges, Computer Science Review 48 (2023) 100549.
[3] D. Alsadie, A comprehensive review of ai techniques for resource management in fog computing: Trends, challenges and future directions, IEEE Access.
[4] B. Mikavica, A. Kostic-Ljubisavljevic, D. Perakovic, I. Cvitic, Deadline-aware task offloading and resource allocation in a secure fog-cloud environment, Mobile Networks and Applications 29 (1) (2024) 133–146.
[5] N. K. Sehgal, P. C. P. Bhatt, J. M. Acken, Cloud computing with security and scalability, Springer, 2020.
[6] H. Liu, Z. Niu, J. Du, X. Lin, Genetic algorithm for delay efficient computation offloading in dispersed computing, Ad Hoc Networks 142 (2023) 103109.
[7] G. Agarwal, S. Gupta, R. Ahuja, A. K. Rai, Multiprocessor task scheduling using multi-objective hybrid genetic algorithm in fog–cloud computing, Knowledge-Based Systems 272 (2023) 110563.
[8] D. Baburao, T. Pavankumar, C. Prabhu, Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method, Applied Nanoscience 13 (2) (2023) 1045–1054.
[9] S. D. Vispute, P. Vashisht, Energy-efficient task scheduling in fog computing based on particle swarm optimization, SN computer science 4 (4) (2023) 391.
[10] M. K. Alasmari, S. S. Alwakeel, Y. A. Alohali, A multi-classifiers based algorithm for energy efficient tasks offloading in fog computing, Sensors 23 (16) (2023) 7209.
[11] G. Goel, A. K. Chaturvedi, A systematic review of task offloading & load balancing methods in a fog computing environment: Major highlights & research areas, in: 2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT), IEEE, 2023, pp. 1–5.
[12] Z. Wei, B. Li, R. Zhang, X. Cheng, L. Yang, Many-to-many task offloading in vehicular fog computing: A multi-agent deep reinforcement learning approach, IEEE Transactions on Mobile Computing 23 (3) (2023) 2107–2122.
[13] Q. Wu, S. Wang, H. Ge, P. Fan, Q. Fan, K. B. Letaief, Delay-sensitive task offloading in vehicular fog computing-assisted platoons, IEEE Transactions on Network and Service Management 21 (2) (2023) 2012–2026.
[14] H. Tran-Dang, D.-S. Kim, Dynamic collaborative task offloading for delay minimization in the heterogeneous fog computing systems, journal of communications and networks 25 (2) (2023) 244–252.
[15] B. Premalatha, P. Prakasam, Optimal energy-efficient resource allocation and fault tolerance scheme for task offloading in iot-fog computing networks, Computer networks 238 (2024) 110080.
[16] M. R. Rezaee, N. A. W. A. Hamid, M. Hussin, Z. A. Zukarnain, Fog offloading and task management in iot-fog-cloud environment: Review of algorithms, networks and sdn application., IEEE Access.
[17] V. Jain, B. Kumar, Qos-aware task offloading in fog environment using multi-agent deep reinforcement learning, Journal of Network and Systems Management 31 (1) (2023) 7.

[18] A. Mahapatra, K. Mishra, R. Pradhan, S. K. Majhi, Next generation task offloading techniques in evolving computing paradigms: Comparative analysis, current challenges, and future research perspectives, Archives of Computational Methods in Engineering 31 (3) (2024) 1405–1474.
[19] O. Akyıldız, İ. Kök, F. Y. Okay, S. Özdemir, A p4-assisted task offloading scheme for fog networks: an intelligent transportation system scenario, Internet of Things 22 (2023) 100695.
[20] B. B. Dash, S. S. Patra, R. Satpathy, B. Dash, Improvement of sdn-based task offloading using golden jackal optimization in fog center, in: 2023 world conference on communication & computing (WCONF), IEEE, 2023, pp. 1–6.
[21] B. Kar, W. Yahya, Y.-D. Lin, A. Ali, Offloading using traditional optimization and machine learning in federated cloud–edge–fog systems: A survey, IEEE Communications Surveys & Tutorials 25 (2) (2023) 1199–1226.
[22] M. Y. Akhlaqi, Z. B. M. Hanapi, Task offloading paradigm in mobile edge computing-current issues, adopted approaches, and future directions, Journal of Network and Computer Applications 212 (2023) 103568.
[23] A. Khiat, M. Haddadi, N. Bahnes, Genetic-based algorithm for task scheduling in fog–cloud environment, Journal of Network and Systems Management 32 (1) (2024) 3.
[24] F. A. Saif, R. Latip, Z. M. Hanapi, K. Shafinah, Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing, IEEE Access 11 (2023) 20635–20646.
[25] K. M. Matrouk, A. D. Matrouk, Mobility aware-task scheduling and virtual fog for offloading in iot-fog-cloud environment, Wireless Personal Communications 130 (2) (2023) 801–836.
[26] S. O. Ogundoyin, I. A. Kamil, Optimal fog node selection based on hybrid particle swarm optimization and firefly algorithm in dynamic fog computing services, Engineering Applications of Artificial Intelligence 121 (2023) 105998.
[27] F. A. Saif, R. Latip, Z. M. Hanapi, M. A. Alrshah, S. Kamarudin, Workload allocation toward energy consumption-delay trade-off in cloud-fog computing using multi-objective npso algorithm, IEEE Access 11 (2023) 45393–45404.
[28] S. Mirjalili, S. Mirjalili, Genetic algorithm, Evolutionary algorithms and neural networks: Theory and applications (2019) 43–55.
[29] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft computing 22 (2) (2018) 387–408.