Comparative Analysis of Nature-Inspired Algorithms for Energy Efficiency and Load-Balancing in the Edge-Cloud Environment

Kevin Afachao[1] , Adnan M. Abu-Mahfouz[1,2], Gerhard P. Hancke[1,3]
[1]*Department of Electrical, Electronic and Computer Engineering*
University of Pretoria
Pretoria, 0028, South Africa
u22851217@tuks.co.za
[2]*Council for Scientific and Industrial Research (CSIR)*
Pretoria, 0083, South Africa
a.abumahfouz@ieee.org
[3]*Department of Computer Science, City University of Hong Kong, Hong Kong SAR*
Pokfulam, Hong Kong
ghancke@ieee.org

Abstract— **This paper presents a comprehensive analysis of nature-inspired metaheuristic algorithms for achieving energy efficiency in the Edge-Cloud environment. The study focuses on the Particle Swarm Algorithm (PSO), Ant Colony Optimization (ACO), and Firefly algorithm, evaluating their performance in workload distribution balance, processing speed, and energy consumption. The simulations are conducted using the ReCloud Simulator. The results reveal that the PSO algorithm outperforms the ACO and Firefly algorithms in workload distribution balance. The ACO algorithm excels in exploration, while the Firefly algorithm demonstrates superior processing speed. However, the Firefly algorithm exhibits slight performance variations due to its sensitivity to workload characteristics. Both the Firefly and PSO algorithms show energy efficiency comparable to or slightly lower than the ACO algorithm. These findings contribute to a better understanding of the strengths and weaknesses of each algorithm, offering valuable insights for researchers and practitioners in the field of energy-efficient computation offloading in the Edge-Cloud environment.**

Keywords— **edge computing, energy-efficiency, metaheuristic algorithm, optimization algorithm,**

I. INTRODUCTION

The Internet of Things (IoT) generates a vast amount of data each year, overwhelming current architectures like Cloud Computing due to high latency and bandwidth costs associated with data transmissions. However, advancements in Edge computing infrastructure, such as improved processing units (including neuromorphic processors), increased storage capacity, and enhanced cyber-physical security architectures [1], [2], have made the network edge an attractive option for offloading computation tasks from mobile devices. This is particularly crucial for mobile devices as conventional IoT applications, with their power-hungry and latency-sensitive characteristics, lead to excessive energy consumption and heat generation. Consequently, battery lifespan is compromised, and system performance suffers. To address these issues and ensure energy efficiency, computation offloading to local edge servers has gained prominence. Nevertheless, the heterogeneous nature of the Edge-IoT environment introduces numerous conflicting objectives and trade-offs. Privacy versus convenience, customization versus security, and complexity versus tractability are all factors that need to be considered. While various algorithms have been developed to strike a balance, most of them are impractical due to constraints related to response time and Quality of Service (QoS). [3] In response to this challenge, researchers have proposed nature-inspired metaheuristic algorithms, which provide optimal solutions to multi-objective problems within limited timeframes without compromising QoS. Unlike heuristic algorithms, mathematical optimization, or machine learning methods, nature-inspired algorithms excel in scalability, tractability, efficiency, and simplicity of search space optimization. [3]

Among these algorithms, Swarm Intelligence stands out as a promising approach. Inspired by the collaborative behaviour of social creatures like bees and ants, Swarm Intelligence enables adaptation to environmental changes while staying focused on objectives. [4] Leveraging the advantages of nature-inspired algorithms, Swarm Intelligence, and other similar metaheuristic algorithms have gained attention as effective solutions for energy efficiency in Edge environments. Despite the extensive coverage of nature-inspired algorithms for energy efficiency in Edge environments, there is a research gap in terms of comparative performance analysis specifically evaluating the Particle Swarm Algorithm, Ant Colony Optimization, and Firefly algorithm. These algorithms have been widely studied and applied in various domains like the Edge. [5]

This paper aims to fill this research gap by comparing these algorithms comprehensively, with a specific focus on their performance in achieving energy efficiency and addressing load-balancing and latency constraints through computation offloading in the Edge-Cloud environment. Through this analysis, the paper aims to identify the strengths and weaknesses of each algorithm, providing valuable insights for researchers and practitioners in the field. The simulations are performed using the ReCloud Simulator to ensure accurate evaluation and comparison.

The rest of the paper is organized as follows: Section II provides the background, followed by a discussion of nature-inspired algorithms for energy efficiency at the Edge in Section III. Section IV presents the simulations conducted, and Section V explores the challenges in research. Finally, the paper concludes with Section VI.

II. BACKGROUND

Energy utilization and task distribution in Edge-Cloud environments have been the focus of various studies, with computation offloading being a commonly employed approach. Rodrigues et al.[6] proposed the use of the Particle Swarm Optimization (PSO) algorithm to achieve low execution time by considering transmission and processing delays in distributed Edge-Cloud infrastructure. However, Huang et al.[7] extended Rodrigues et al.'s research by incorporating energy cost and security risks into the offloading process, and they opted for the Ant Colony Optimization (ACO) algorithm over PSO. They argued that ACO has a better ability to find global optima for multi-objective problems, while PSO exhibits a weaker exploration mechanism compared to exploitation. Huang et al. introduced two algorithms, EA-RMIP and EA-OMIP, by combining ACO with Mixed Integer Programming (MIP), resulting in an energy-efficient offloading strategy that accounts for security considerations. Comparative analysis showed that EA-RMIP outperformed EA-OMIP in terms of convergence time. Whereas EA-OMIP was designed using conventional MIP techniques, EA-RMIP is an adaptation of a hybrid technique based on recent studies.

Recognizing the potential for further enhancements, researchers have explored hybrid approaches by combining PSO and ACO to achieve improved results compared to individual algorithms. Pawar et al.[8] discovered that traditional load-balancing algorithms like First Come First Serve were inefficient for growing numbers of cloud service users. As a solution, they introduced the Ant Colony Honey Bee Dynamic Feedback (ACHBDF) algorithm, which combines ACO and the Honey Bee algorithm for resource balancing in Cloud Computing. Zahoor et al.[9] conducted similar research and compared various load balancing approaches, including PSO, Throttle, and Round Robin, in terms of latency performance. They found that PSO outperformed the other algorithms. Building on this hybrid metaheuristic approach, Zahoor et al. later combined Ant Colony Optimization (ACO) with Ant Bee Colony (ABC) algorithms, resulting in the Hybrid Artificial Bee Ant Colony Optimization (HABACO) algorithm for load balancing[10]. HABACO demonstrated superior performance in terms of average processing time and response time compared to ACO, PSO, and ABC algorithms. Other studies, such as Liu et al.[11] and Bi et al.[12], have also successfully utilized hybrid algorithms combining ACO with Fuzzy Clustering (FC), Genetic Algorithm (GA), Simulated Annealing (SA), and PSO, respectively, to address specific task scheduling and offloading challenges[13], [14].

In contrast, despite its potential for improving energy efficiency, the Firefly algorithm lacks extensive research highlighting its strengths. Adhikari et al.[15] introduced the Firefly algorithm, inspired by fireflies' mating rituals, for optimizing the placement of application tasks. The algorithm outperformed benchmarks in terms of computational time, energy consumption, CO2 emission, and temperature emission.

III. NATURE-INSPIRED ALGORITHMS

Nature-inspired metaheuristic algorithms have gained significant attention in solving optimization problems due to their ability to mimic natural phenomena and provide efficient solutions. This section discusses three prominent nature-inspired metaheuristic algorithms: Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and the Firefly Algorithm. We examine their concepts, strengths, weaknesses, and relevance to computation offloading and energy efficiency in the IoT environment.

A. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a population-based optimization algorithm that simulates the collective behaviour of bird flocking or fish schooling[16]. It utilizes particles representing potential solutions, which iteratively adjust their positions based on their own experiences and the experiences of neighbouring particles. The strengths of PSO lie in its simplicity and ease of implementation, making it easily understandable and applicable. It demonstrates good exploration capabilities, enabling efficient search across a wide solution space. PSO has been successfully applied to various domains, particularly for continuous optimization problems. However, converging to the global optimum in complex, multimodal optimization problems may struggle. Premature convergence, where particles converge to suboptimal solutions too quickly, can limit its effectiveness. The performance of PSO heavily relies on parameter tuning, posing challenges in selecting appropriate values. In the context of computation offloading and energy efficiency in the IoT environment, PSO has been utilized for optimizing task allocation and resource utilization. Its exploration capabilities can aid in finding efficient offloading strategies. However, limitations such as premature convergence may restrict its effectiveness in dynamic IoT scenarios.

B. Ant Colony Optimization (ACO)

The Ant Colony Optimization (ACO) is a metaheuristic algorithm inspired by the foraging behaviour of ants[17]. It employs pheromone trails to guide the search for optimal solutions. Ants deposit pheromones on their paths, and other ants follow these trails, favouring paths with higher pheromone concentration. ACO excels in solving combinatorial optimization problems with discrete decision variables. It efficiently handles multi-objective optimization, making it suitable for complex, conflicting objectives. ACO exhibits adaptability and performs well in dynamic environments due to its distributed nature. However, ACO can suffer from slow convergence, particularly in large-scale problems. The algorithm's performance heavily relies on parameter tuning, which can be time-consuming and require domain expertise. ACO may struggle to find optimal solutions in high-dimensional search spaces. In the context of computation offloading and energy efficiency in the IoT environment, ACO has been employed for optimizing task allocation, resource utilization, and load balancing. Its ability to handle multi-objective optimization addresses the trade-off between energy efficiency, response time, and other performance metrics. However, consideration should be given to ACO's convergence speed and sensitivity to parameter settings in dynamic and resource-constrained IoT scenarios.

C. Firefly Algorithm

The Firefly Algorithm is inspired by the flashing behaviour of fireflies. It imitates the social behaviour of fireflies and their attraction towards brighter individuals[4]. The algorithm utilizes attractiveness between fireflies to guide the search for

optimal solutions. The Firefly Algorithm demonstrates fast convergence speed, making it suitable for optimization problems with limited computation resources. It performs well in continuous optimization problems and has been successfully applied to the cloud domain[18]. The algorithm shows potential for improving energy efficiency due to its quick convergence to near-optimal solutions. However, it may encounter challenges in solving large-scale and complex optimization problems. Its exploration capabilities may be limited, potentially leading to premature convergence. Further research is needed to explore its effectiveness in dynamic environments and multi-objective optimization. In the context of computation offloading and energy efficiency in the IoT environment, the Firefly Algorithm can be employed for optimizing resource allocation and energy efficiency. Its fast convergence speed enables real-time decision-making and efficient resource utilization. However, its limitations in handling complex optimization problems and exploration capabilities should be considered in dynamic IoT scenarios.

IV. SIMULATIONS

A. Key Parameters

1) Degree of Imbalance: The degree of imbalance metric reflects the extent of workload distribution imbalance among the servers in the system.

2) Processing Speed: The processing speed metric reflects the time required to execute the system's computation tasks.

3) Energy consumption: Energy consumption is a crucial metric in evaluating the efficiency and sustainability of the algorithms. The energy consumption model depends on CPU utilization during system processes such as busy, idle, and task offloading.

B. Simulation Workflow

The simulation in this study aims to replicate a real-time environment by modelling devices, logical components, and management policies. To facilitate the simulation, the CloudSim simulator was utilized. Additionally, a supplementary model called ReCloud was employed to compare different algorithms within the Edge-Cloud simulation context. The simulation process involves several steps, including initializing ReCloud, setting up cloud servers, creating cloudlets, brokers, virtual machines (VMs), tasks, and specifying scheduling algorithms. The environment was then simulated, and the results were monitored. The ReCloud library enables the comparison of scheduling algorithms initiated within the broker. It is important to note that while ReCloud is an extension of CloudSim, it does not come preconfigured with components such as sensors and actuators.

The performance of the three algorithms was evaluated three times in the context of these parameter settings. PSO was configured with the following parameter settings: p = 100, iterations = 100, minInertia = 0.4, maxInertia = 0.9, c1 = 1.5, c2 = 1.5, k = 5. These parameters govern the behaviour of the PSO algorithm, including the number of particles, the number of iterations, the inertia range, the cognitive and social coefficients, the constriction factor, and the methods used for updating inertia and particle position. ACO employs the following parameter values: ants = 100, iterations = 100, initial-Pheromone = 0.001, alpha = 1, beta = 4, q = 1, and rho = 0.1. These parameters control the behaviour of the ACO algorithm, such as the number of ants, the number of iterations, the importance of pheromones, and the influence of distance priority. Firefly Algorithm utilized the following parameter values: numFireflies = 100, maxIterations = 100, alpha = 0.1, beta = 0.9, and gamma = 0.5. These parameters influence the behaviour of the firefly algorithm, such as the number of fireflies, the maximum number of iterations, the attractiveness and randomness factors, and the light absorption factor.

The parameter selection for the PSO algorithm was based on the trade-off between exploration and exploitation. The number of particles (p) determines the population size, with larger populations increasing computation overhead but smaller populations risking premature convergence. The number of iterations (iter) defines the stopping criterion, typically ranging between 100 and 1000 depending on the problem's complexity. A value of 100 was chosen to strike a balance between exploration and convergence. The inertia weight (min Inertia, max Inertia) controls global and local exploration, with higher values encouraging exploration and lower values promoting exploitation. Cognitive and Social coefficients (c1, c2) influence velocity and position updates, impacting exploration and exploitation capabilities. The constriction factor (k) ensures convergence by reducing velocity near the best positions. In the ACO algorithm, the number of ants (ants) and iterations (iterations) determine population size and stopping criterion, respectively. The initial pheromone level (initial-Pheromone) influences early-stage path preference and is set to small positive values to prevent premature convergence. Pheromone importance (alpha) and distance priority (beta) balance exploitation and exploration. Total pheromone (q) controls the strength of pheromone updates, with higher values favouring exploitation. Pheromone vaporization (rho) regulates the rate of pheromone evaporation, with higher values promoting more exploration. In the Firefly algorithm, the number of fireflies sets the population size and maximum iterations for the stopping criterion. The attraction coefficients alpha, beta, and gamma govern fireflies' movement and attraction behaviour, with fine-tuning necessary to balance exploration and exploitation.

These parameter choices were made to optimize the algorithms' performance based on their respective characteristics and the desired exploration-exploitation trade-off[3].

C. Environment Setup

In the experimental setup, the environment consists of two servers utilizing x86 architecture and employing Xen as the Virtual Machine (VM) monitor. Each server is equipped with three host devices, each offering a processing capacity of 177730 MIPS and comprising six processing elements. The servers are configured with a RAM memory capacity of 16000 MB, a bandwidth of 1500 MB/s, and a storage capacity of 4000 GB. Additionally, the Edge-Cloud environment simulated includes two Brokers and three Virtual Machines (VMs).

V. RESULTS

The results of the experiment, as shown in Fig.1, Fig. 2 and Fig. 3 provide insights into the performance of the PSO, ACO,

and Firefly algorithms in terms of the degree of imbalance, processing speed and energy consumption.

The experiment results for the PSO algorithm demonstrated a degree of imbalance of 2.3154, 2.5831 and 2.5705 in the first, second, and third scenarios, respectively. The processing speeds were 780 ms, 644 ms, and 1384 ms, while the energy consumption values were 24.524 KJ, 21.798 KJ and 21.266 KJ for the first, second and third scenarios, respectively. The ACO algorithm showed a degree of imbalance of 2.8941, 2.5835 and 2.5587 in the first, second and third scenarios, respectively. For processing speed, the ACO algorithm achieved 738 ms, 534 ms, and 1204 ms in the first, second, and third scenarios, respectively. The corresponding energy consumption values were 24.421 KJ, 21.490 KJ. The Firefly algorithm yielded a degree of imbalance of 2.861, 2.5469 and 2.5341 in the first, second, and third scenarios, respectively. In terms of processing speed, the Firefly algorithm achieved 88 ms, 178 ms, and 214 ms, with energy consumption values of 24.420 KJ, 21.593 KJ, and 21.232 KJ for the first, second, and third scenarios, respectively.

The PSO algorithm's results indicate that the PSO algorithm achieved a lower degree of imbalance compared to both the ACO and Firefly algorithms. The strength of the PSO algorithm lies in its ability to strike a balance between exploration and exploitation, allowing it to converge towards a near-optimal solution. The weakness of the PSO algorithm may be its limited ability to escape local optima, which can result in suboptimal workload distribution. In contrast, the ACO algorithm exhibited a relatively higher degree of imbalance compared to the other algorithms in all three scenarios. The strength of the ACO algorithm lies in its ability to explore the solution space and find diverse solutions. However, this characteristic may lead to a less optimal workload distribution, resulting in a higher degree of imbalance. One possible weakness of the ACO algorithm is its sensitivity to parameter settings, such as the initial pheromone level and the pheromone importance, which may require careful tuning for optimal results The Firefly algorithm produced a degree of imbalance similar to the ACO algorithm, but slightly higher than the PSO algorithm. The Firefly algorithm's strength lies in exploiting solutions efficiently and converging towards optimal workload distribution. However, one potential weakness of the Firefly algorithm is its sensitivity to parameter settings, particularly the attraction coefficients, which may require careful tuning for optimal performance.

In the experiments, the PSO algorithm demonstrated competitive processing speeds, particularly in the second scenario. This highlights the strength of the PSO algorithm in achieving efficient task allocation and execution. However, in the third scenario, the PSO algorithm exhibited slower processing speeds compared to the other algorithms, indicating a potential weakness in handling more complex scenarios with larger computation loads. The ACO algorithm demonstrated relatively fast processing speeds, particularly in the second scenario. This indicates the strength of the ACO algorithm in efficiently allocating computation tasks among the available resources. However, it is worth noting that the processing speeds achieved by the ACO algorithm were slightly higher compared to the other algorithms in all three scenarios. This suggests a potential weakness of the ACO algorithm in terms of computational efficiency. The Firefly algorithm demonstrated significantly faster processing speeds than the ACO and PSO algorithms in all three scenarios. This highlights the strength of the Firefly algorithm in achieving high computational efficiency and fast task execution. However, it is important to note that the Firefly algorithm exhibited slightly slower processing speeds compared to the other algorithms in the first and second scenarios, suggesting a potential weakness in certain scenarios with specific workload characteristics.

In the experiments, the PSO algorithm demonstrated competitive energy consumption, particularly in the second and third scenarios. This highlights the strength of the PSO algorithm in achieving energy-efficient task allocation and execution. However, in the first scenario, the PSO algorithm consumed slightly higher energy than the other algorithms, suggesting a potential weakness in certain scenarios. The ACO algorithm demonstrated relatively consistent energy consumption across the scenarios, indicating its stability and effectiveness in managing energy resources. However, it is worth noting that the ACO algorithm consumed slightly higher energy compared to the Firefly and PSO algorithms in all three scenarios. This suggests a potential weakness of the ACO algorithm in terms of energy efficiency. The Firefly algorithm exhibited comparable energy consumption to the other algorithms, demonstrating its ability to achieve energy-efficient task allocation and execution. However, in the first scenario, the Firefly algorithm consumed slightly higher energy compared to the other algorithms, indicating a potential weakness in specific scenarios.

Overall, the PSO algorithm demonstrated superior workload distribution balance compared to the ACO and Firefly algorithms, with a 95% confidence interval ranging from 2.47 to 2.74. While the ACO algorithm excelled in exploration and the Firefly algorithm showed strength in exploitation, both may benefit from parameter fine-tuning to improve workload distribution. The Firefly algorithm exhibited faster processing speed, outperforming the ACO and PSO algorithms in most scenarios, with a 95% confidence interval between 295.43 and 985.46. However, it also displayed slight performance variations, indicating sensitivity to workload characteristics. In terms of energy consumption, the Firefly and PSO algorithms showed comparable or slightly lower energy usage compared to the ACO algorithm, with a 95% confidence interval for the true population mean falling within the range of 21.25 and 23.6, highlighting their energy efficiency. All three algorithms effectively managed energy resources.

VI. CHALLENGES

A. Lack of Scalability

Nature-inspired algorithms, such as PSO, ACO, and the Firefly Algorithm, may encounter scalability issues when dealing with large-scale problems. The increasing complexity and dimensionality of optimization problems in the IoT environment pose challenges for these algorithms to maintain their effectiveness and efficiency.

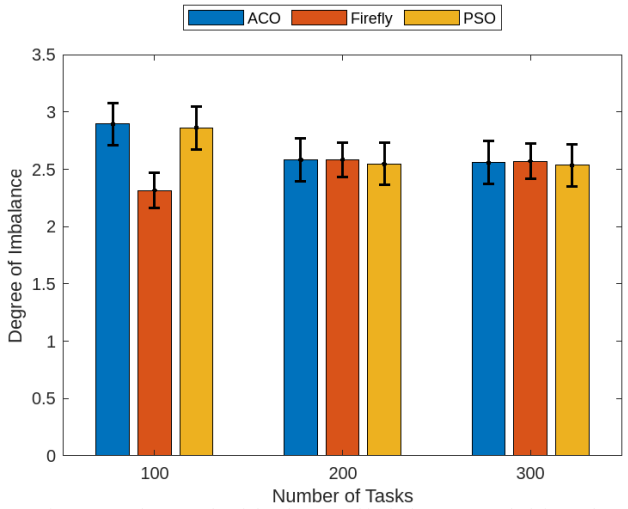

Figure 1: A bar graph of the degree of imbalance recorded from the algorithms.

B. *Premature Convergence*

Premature convergence is a common limitation in nature-inspired algorithms, where the search prematurely converges to suboptimal solutions without exploring the entire solution space. This can lead to the algorithms getting trapped in local optima and failing to find globally optimal solutions.

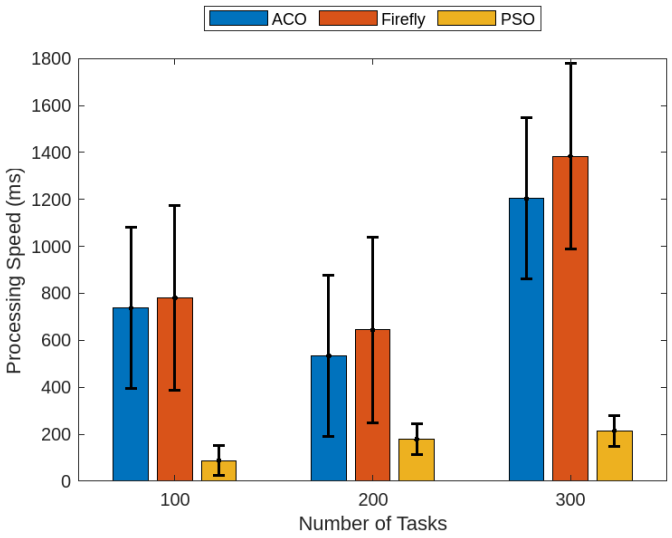

Figure 2: A bar graph of the processing speed recorded in milliseconds from the algorithms.

C. *Parameter Tuning Sensitivity*

Nature-inspired algorithms often rely on a set of parameters that require careful tuning to achieve optimal performance. Selecting appropriate parameter values can be time-consuming and challenging, as different problem domains and scenarios may require specific parameter configurations for achieving desirable results.

D. *Limited Handling of Dynamic Environments*

Dynamic environments in the IoT context involve changing conditions, such as fluctuating workloads, resource availability, and network connectivity. Nature-inspired algorithms may struggle to adapt to dynamic environments due to their inherent design and the challenges of maintaining a balance between exploration and exploitation in real-time scenarios.

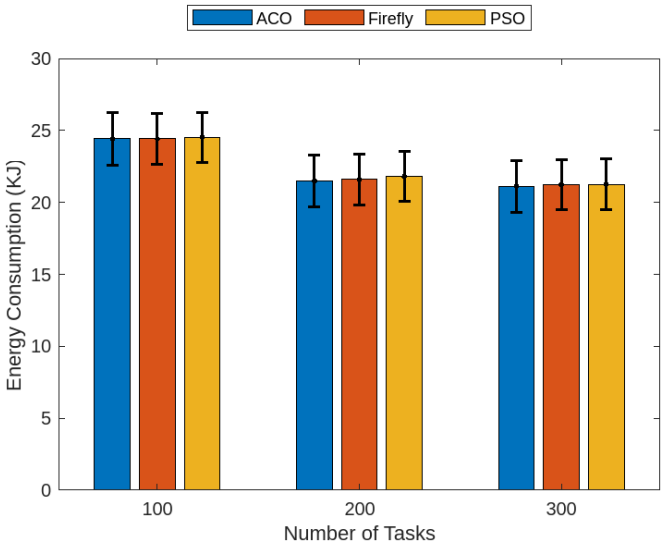

Figure 3: A bar graph of the algorithms' energy consumption recorded in milliseconds.

E. *Lack of Guarantee for Optimality*

While nature-inspired algorithms aim to find near-optimal solutions, they do not provide guarantees for achieving global optimality. The stochastic nature of these algorithms and their reliance on exploration make it challenging to ensure finding the absolute best solution for complex optimization problems.

VII. CONCLUSION

In this study, we conducted a comparative analysis of nature-inspired metaheuristic algorithms, namely the PSO, ACO, and Firefly algorithms, in the context of energy efficiency and load-balancing for computation offloading in the Edge-Cloud environment. The results demonstrate the strengths and weaknesses of each algorithm, providing valuable insights for decision-making.

The PSO algorithm excelled in workload distribution balance, outperforming ACO and Firefly. ACO showcased exploration capabilities, while Firefly demonstrated superior processing speed. Both Firefly and PSO algorithms exhibited comparable or slightly lower energy consumption than ACO, indicating their effectiveness in energy efficiency.

Overall, this research provides a comprehensive understanding of the performance and characteristics of nature-inspired metaheuristic algorithms in the context of energy efficiency and load-balancing for computation offloading in the Edge-Cloud environment. These findings can guide researchers and practitioners in selecting the most suitable algorithm based on specific requirements and objectives. Future research would focus on further enhancing the algorithms' performance by varying the specific parameter values since the current research is limited to specific parameter values. Optimization techniques would be utilized in addressing the identified weaknesses to maximize their potential in energy-efficient computation offloading.

ACKNOWLEDGEMENT

This work is based on the research supported in part by our industry partner Telkom.

REFERENCES

[1] F. Saeik *et al.*, "Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions," *Computer Networks*, vol. 195. Elsevier B.V., Aug. 04, 2021. doi: 10.1016/j.comnet.2021.108177.

[2] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2. Institute of Electrical and Electronics Engineers Inc., pp. 869–904, Oct. 2020. doi: 10.1109/COMST.2020.2970550.

[3] M. Adhikari, S. N. Srirama, and T. Amgoth, "A comprehensive survey on nature-inspired algorithms and their applications in edge computing: Challenges and future directions," *Softw Pract Exp*, vol. 52, no. 4, pp. 1004–1034, Apr. 2022, doi: 10.1002/spe.3025.

[4] X.-S. Yang, S. Deb, S. Fong, X. He, and Y.-X. Zhao, "From Swarm Intelligence to Metaheuristics Nature Inspired Optimization Algorithms," *Computer*, pp. 52–59, 2016. doi: 10.1109/MC.2016.292.

[5] D. Kumar, G. Baranwal, Y. Shankar, and D. P. Vidyarthi, "A survey on nature-inspired techniques for computation offloading and service placement in emerging edge technologies," *World Wide Web*, vol. 25, no. 5, pp. 2049–2107, Sep. 2022, doi: 10.1007/s11280-022-01053-y.

[6] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "A PSO Model with VM Migration and Transmission Power Control for Low Service Delay in the Multiple Cloudlets ECC Scenario," in *International Conference on Communications*, Paris: IEEE, May 2017. doi: 10.1109/ICC.2017.7996358.

[7] X. Huang, Y. Yang, and X. Wu, "A Meta-Heuristic Computation Offloading Strategy for IoT Applications in an Edge-Cloud Framework," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Sep. 2019. doi: 10.1145/3386164.3390513.

[8] N. Pawar, U. Kumar Lilhore, N. Agrawal, M. Tech Research Scholar, and A. Professor, "A Hybrid ACHBDF Load Balancing Method for Optimum Resource Utilization In Cloud Computing," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2017 IJSRCSEIT*, vol. 6, no. 10, pp. 367–373, 2017, [Online]. Available: https://www.researchgate.net/publication/340255668

[9] S. Zahoor, N. Javaid, A. Khan, B. Ruqia, F. J. Muhammad, and M. Zahid, "A Cloud-Fog-Based Smart Grid Model for Efficient Resource Utilization," in *International Wireless Communications & Mobile Computing Conference (IWCMC)*, 2018. doi: 10.1109/IWCMC.2018.8450506.

[10] S. Zahoor, S. Javaid, N. Javaid, M. Ashraf, F. Ishmanov, and M. K. Afzal, "Cloud-fog-based smart grid model for efficient resource management," *Sustainability (Switzerland)*, vol. 10, no. 6, Jun. 2018, doi: 10.3390/su10062079.

[11] J. Liu, X. Wei, T. Wang, and J. Wang, "An Ant Colony Optimization Fuzzy Clustering Task Scheduling Algorithm in Mobile Edge Computing," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, Springer Verlag, 2019, pp. 615–624. doi: 10.1007/978-3-030-21373-2_51.

[12] J. Bi, H. Yuan, K. Zhang, and M. C. Zhou, "Energy-Minimized Partial Computation Offloading for Delay-Sensitive Applications in Heterogeneous Edge Networks," *IEEE Trans Emerg Top Comput*, vol. 10, no. 4, pp. 1941–1954, Oct. 2022, doi: 10.1109/TETC.2021.3137980.

[13] Y. Zhang, Y. Liu, J. Zhou, J. Sun, and K. Li, "Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing," *Future Generation Computer Systems*, vol. 112, pp. 148–161, Nov. 2020, doi: 10.1016/j.future.2020.05.025.

[14] L. N. T. Huynh, Q. V. Pham, X. Q. Pham, T. D. T. Nguyen, M. D. Hossain, and E. N. Huh, "Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach," *Applied Sciences (Switzerland)*, vol. 10, no. 1, Jan. 2020, doi: 10.3390/app10010203.

[15] M. Adhikari and H. Gianey, "Energy efficient offloading strategy in fog-cloud environment for IoT applications," *Internet of Things*, vol. 6, 2019, doi: 10.1016/j.iot.2019.10.

[16] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *International Conference on Neural Networks*, Perth: IEEE, 1995, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.

[17] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor Comput Sci*, vol. 344, no. 2–3, pp. 243–278, Nov. 2005, doi: 10.1016/j.tcs.2005.05.020.

[18] N. Kaur and A. Chhabra, "Analytical review of three latest nature-inspired algorithms for scheduling in clouds," in *International Conference on Electrical, Electronics, and Optimization Techniques*, Chennai: IEEE, Mar. 2016. doi: 10.1109/ICEEOT.2016.7755315.

Kevin Afachao (Non-Member, IEEE**)** He holds a BSc in Telecommunications Engineering from the Kwame Nkrumah University of Science and Technology (2019) and a B. Honours degree in Computer Engineering from the University of Pretoria (2022). He is currently pursuing an M. Eng in Computer Engineering at the University of Pretoria, with research interests in the Internet of Things, Edge Computing, and Artificial Intelligence.

Adnan M. Abu-Mahfouz (Senior Member, IEEE) received the M.Eng. and PhD degrees in computer engineering from the University of Pretoria, Pretoria, South Africa, in 2005 and 2011, respectively. He is currently a Chief Researcher and the Centre Manager of the Emerging Digital Technologies for 4IR (EDT4IR) Research Centre, Council for Scientific and Industrial Research, Pretoria; an Extraordinary Professor with the University of Pretoria; a Professor Extraordinaire with the Tshwane University of Technology, Pretoria. His research interests are wireless sensor and actuator networks, low power wide area networks, software-defined wireless sensor networks, cognitive radio, network security, network management, and sensor/actuator node development. Prof Abu-Mahfouz is a Section Editor-in-Chief with the Journal of Sensor and Actuator Networks, an Associate Editor at IEEE INTERNET OF THINGS, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON CYBERNETICS and IEEE ACCESS, and a member of many IEEE technical communities.

Gerhard Hancke (Fellow, IEEE) is a Professor in the Department of Computer Science at the City University of Hong Kong. He received B. Eng and M. Eng degrees in Computer Engineering from the University of Pretoria, South Africa, in 2002 and 2003, and a PhD in Computer Science from the University of Cambridge, United Kingdom, in 2009. Previously he worked as a researcher with the Smart Card and IoT Security Centre and as a teaching fellow with the Department of Information Security, both located at Royal Holloway, University of London. His research interests are system security, reliable communication and distributed sensing for the industrial Internet-of-Things.