Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 44 (2023), No. 6, pp. 1167–1177
DOI : 10.47974/JIOS-1455

Enhancing Quality of Service (QoS) and minimizing application placement delay in cloud-fog nodes through meta-heuristic algorithms

Y. Nasir Ahmed *
S. Pakkir Mohideen †
Department of Computer Applications
B. S. Abdur Rahman Crescent Institute of Science and Technology
Vandalur, Chennai 600048
Tamil Nadu
India

Mohammad Pasha §
Department of Information Technology
Muffakham Jah College of Engineering and Technology
Hyderabad 500034
Telangana
India

Abstract

Efficient module placement in Cloud-Fog computing enhances QoS and minimizes delay. Binary Particle Swarm Optimization (BPSO) and Binary Bat Algorithm (BBA), effective meta-heuristic algorithms, address dynamic fog complexities. This study models placement as binary optimization, incorporating resource availability and latency. BPSO and BBA optimize module placement, demonstrated through simulations improving response time, throughput, and resource utilization. These meta-heuristic methods surpass traditional techniques, optimizing cloud-fog systems, achieving superior QoS, and reduced delay.

Subject Classification: *Primary 68T20, Secondary 68M20.*

Keywords: *QoS improvement, Delay minimization, application module placement, Cloud-Fog nodes, Meta-heuristic algorithms, BPSO, BBA.*

* *E-mail:* **nasir_ca_phd_18@crescent.education (Corresponding Author)**
† *E-mail:* **pakirmoitheen@crescent.education**
§ *E-mail:* **mdpasha@mjcollege.ac.in**

1. Introduction

Background: Cloud-fog computing, a versatile technology, transforms industries and fuels innovation. In IoT, it plays a pivotal role by enabling efficient edge data processing, enhancing real-time decision-making, and scalability. In smart cities, it drives real-time monitoring, data analytics, and resource management for applications like traffic control and energy optimization, contributing to sustainable urban development [1-3]. In healthcare, cloud-fog computing facilitates remote patient monitoring and telemedicine, improving response times and saving lives. In IIoT, it boosts industrial efficiency through predictive maintenance and intelligent decision-making. Edge AI benefits from cloud-fog computing, enabling faster inference and privacy-preserving AI applications [4-5]. Additionally, it enhances mobile networks via mobile edge computing, reducing latency and supporting resource-intensive applications.

2. Related Works

The recent studies indicate that meta-heuristic algorithms continue to be a popular and effective approach for solving the task scheduling problem in cloud-fog computing environments.

Several task scheduling algorithms are proposed for optimizing resource allocation in fog computing environments. The Bat Algorithm-based Fog Computing (BAFC) optimizes resource allocation by considering energy consumption and task processing time to minimize completion time and energy usage [6]. Another hybrid algorithm merges Ant Colony Optimization (ACO) and Artificial Bee Colony (ABC) techniques, aiming to minimize makespan and energy consumption in cloud-fog computing [7]. A Bat Algorithm and Improved Ant Colony Optimization (BA-ACO) method addresses resource heterogeneity, task dependencies, and communication latency for similar optimization goals [8]. Firefly Algorithm (FA) is used to minimize makespan and energy consumption by considering communication latency, processing time, and energy consumption [9].

Different studies assess various algorithms, showing their effectiveness. Enhanced versions of algorithms, like ACO and Whale Optimization Algorithm (WOA) [10], are proposed, considering resource heterogeneity, task dependencies, and communication latency [11][12]. Hybrid approaches, such as Bee Colony Optimization (BCO) combined with heuristics, and Modified Imperialist Competitive Algorithm (ICA),

are developed for task scheduling [13]. Multi-objective optimization models are introduced, integrating energy consumption and completion time, with algorithms like genetic, simulated annealing, and evolutionary algorithms addressing QoS metrics and resource utilization [16] [20][22]. A reinforcement learning approach tackles dynamic resource allocation in fog computing systems through a Markov decision process [21]. A proposed multi-objective binary bat algorithm leverages support vector machines to avoid premature convergence, enhancing task scheduling based on service level agreements and load balancing [25]. All these efforts contribute to improved task scheduling, reducing delays, computation, and communication costs in cloud-fog computing environments [23-25].

3. System Model

We explain the necessary context and background information about the problem domain, including the entities, components, and relationships involved in the research study.

Problem Statement: The problem is to find an optimal placement of a set of modules on a hierarchy of multi-level fog nodes, such that the total latency of the placement is minimized. Each module has a set of attributes, including priority, deadline, MIPS (million instructions per second). The objective function to minimize is defined as the sum of the latencies of each module divided by its MIPS.The objective function for the given problem statement can be defined as follows:

Minimize Σ [(latency of module i / MIPS of module i) * (priority of module i / deadline of module i)] for all modules i where,

- Latency of module i represents the time taken for module i to complete its execution
- MIPS of module i represents the processing speed of module i in million instructions per second,
- Priority of module i represents the importance or urgency of module i,
- Deadline of module i represents the maximum allowable time for module i to complete its execution.

The objective function combines latency-to-MIPS and priority-to-deadline ratios for each module, ensuring processing efficiency (MIPS)

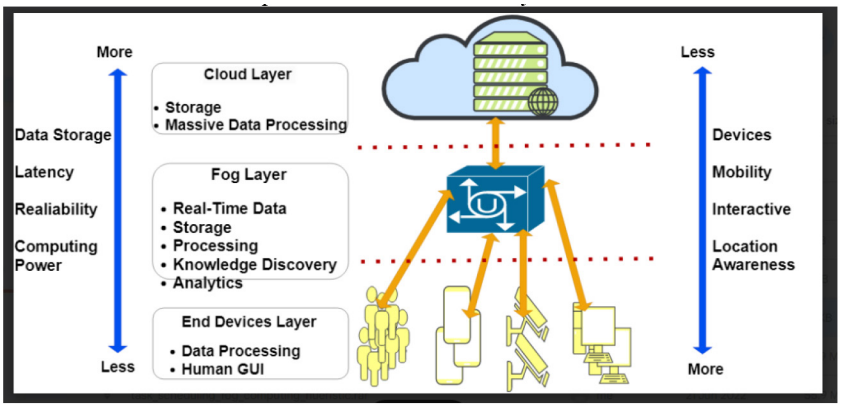

Figure 1

Architecture of Application Module placement in Cloud-Fog Environment

and module importance (priority) and timeliness (deadline) in minimizing overall latency. The Figure 1, shows the architecture of Application Module placement in Cloud-Fog Environment. Assumptions and notations include N modules with different MIPS and deadlines placed on M multi-level fog nodes of varying capacities. In a real-time setup, modules have deadlines, and latency constraints are enforced with a latency model. The goal is to minimize total completion time within latency and deadline constraints.

4. Methodology

Module placement optimizes fog node performance and resource utilization. Crucial in fog computing, it affects system performance, energy, and cost. Several algorithms exist, like Binary Bat Algorithm (BBA), inspired by bats' echolocation. BBA's binary encoding encodes variables as binary strings. By employing BPSO and BBA, the problem models as binary optimization, considering resources, latency, and load balance. They offer optimal mappings to minimize delay and enhance QoS, including response time and throughput.

The Binary Particle Swarm Optimization (BPSO) algorithm operates on binary search spaces. It initializes particles with binary strings and has two phases: initialization (random binary strings) and iteration (updating particles' positions and velocities based on fitness evaluations). It seeks global and local best particles. The velocity and position of each particle are then updated according to the following equations:

$$v(i,j) = w * v(i,j) + c1 * rand() * (pbest(i,j) - x(i,j)) + c2 * rand() * (gbest(j) - x(i,j)) \; ; \; x(i,j) = sigmoid(v(i,j))$$

where:

v(i,j) is the velocity of particle i in dimension j

- w is the inertia weight
- c1 and c2 are the cognitive and social parameters, respectively
- rand() is a random number between 0 and 1
- pbest(i,j) is the best position of particle i in dimension j
- gbest(j) is the best position of the global best particle in dimension j
- x(i,j) is the position of particle i in dimension j
- sigmoid() is a sigmoid function that maps the value of v(i,j) to a binary value of 0 or 1

The iteration phase is repeated unless a stopping criterion is satisfied, such as an utmost amount of iterations or a minimum fitness value. At the end of the algorithm, the best particle found is returned as the solution to the optimization problem.

The Binary Bat Algorithm (BBA): The Binary Bat Algorithm (BBA) is a variant of the traditional Bat Algorithm tailored for binary search spaces. It includes initialization (random binary strings) and iteration (updating bats' positions and velocities based on fitness evaluations). It aims to find global and local best bats.. The velocity and position of each bat are then updated according to the following equations:

- A(i) = Amin + (Amax - Amin) * rand()
- r(i,j) = rand(); v(i,j) = v(i,j) + (x(i,j) - gbest(j)) * A(i)
- x(i,j) = sigmoid(v(i,j))
- if r(i) > r0: ; x(i,j) = local(j)

where:

A(i) is the loudness of bat i

Amin and Amax are the minimum and maximum loudness values, respectively

rand() is a random number between 0 and 1

r(i,j) is a random value for each dimension of bat i

v(i,j) is the velocity of bat i in dimension j

local(j) is the best position of the local best bat in dimension j

sigmoid() is a sigmoid function that maps the value of v(i,j) to a binary value of 0 or 1.

r0 is a random threshold value that determines whether or not a bat will use its local best position

The iteration phase is recurred until a stopping criterion is satisfied, such as an utmost amount of iterations or a minimum fitness value.

4. Implementation

The experiment is written in Java for conceptual proof and IFogSim is used for simulations. It implements two optimization algorithms, Binary Particle Swarm Optimization (BPSO) and Binary Bat Algorithm (BBAT), for solving a module placement problem. The code defines several functions, including the fitness function, which calculates the total latency of a placement, and the initialization functions for the data, particle population, and bat population.

Ensuring QoS by considering module execution deadline and MIPS in IFogSim:

IFogSim ensures QoS via module deadline and MIPS consideration. BPSO and BBAT algorithms require specifying type and parameters in the input file, generating optimized placement for QoS in output. Fog computing QoS relies on latency, bandwidth, resources, and workload. For this, setting up the IFogSim environment and defining fog computing scenarios, including fog nodes, cloud servers, IoT devices, and the network. Specifying the tasks, modules, and their QoS requirements, such as response time and reliability. Now implementing a module placement strategy considering available MIPS and deadlines. Hence fog gateway

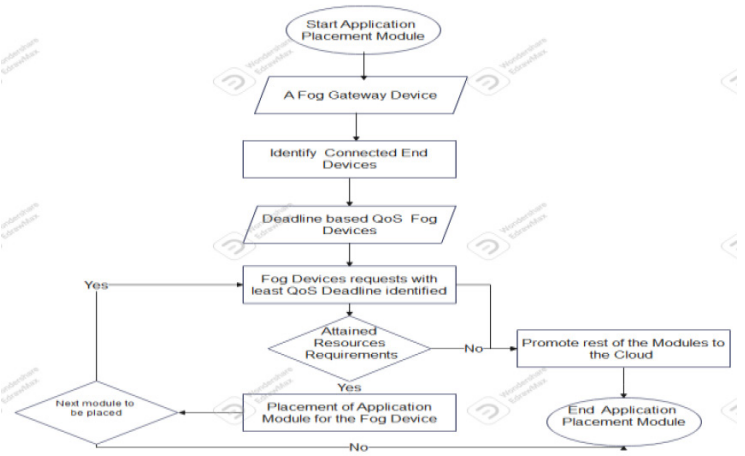

Figure 2

Flowchart-Application Placement module

nodes (1-2), Number of devices(5-20), sensing intervals(5-10 seconds) are used to conduct simulations for performance evaluation, and fine-tune the algorithms for optimal QoS. Monitored the execution progress and implemented dynamic re-scheduling for deadline violations. Therefore this approach ensures that fog computing environment meets desired QoS standards. Flowchart for the application module placement is represented in Figure 2.

5. Results and Discussions

Among the commonly used QoS comparison metrics in cloud-fog computing are response time, throughput, latency, scalability, and resource utilization. To optimize these QoS metrics, metahuristic algorithms such as Particle Swarm Optimization (PSO) and Bat Algorithm are employed to optimize the system's scalability by dynamically adjusting resource allocation to accommodate varying workloads and efficiently manage a growing number of users or devices. By leveraging these advanced optimization techniques, cloud-fog computing can achieve higher levels of efficiency, responsiveness, and adaptability, making it a compelling solution for various applications across diverse industries and also empowers stakeholders to make informed decisions. Below figure represents the execution time (ET) for three algorithms: Greedy, BPSO, and BBAT, considering different configurations of the system with varying numbers of gateways (G), end devices (ED), and sensing intervals (SI).

In Figure 3, we can observe that BBAT consistently outperforms Greedy and BPSO in terms of execution time for all three configurations. In Figure 4. The Greedy algorithm was unable to find a placement for the

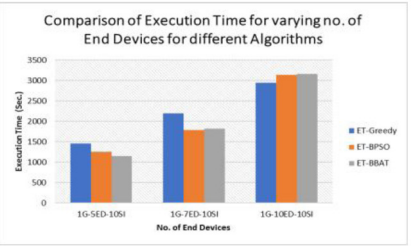

Figure 3

Execution Time (ET) for simulated algorithms: Greedy, BPSO, and BBAT.

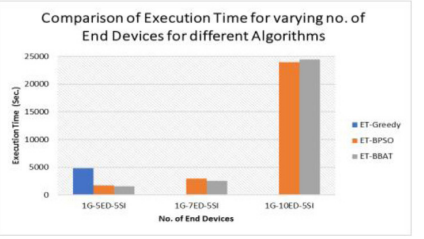

Figure 4

Execution Time (ET) for simulated algorithms: Greedy, BPSO, and BBAT

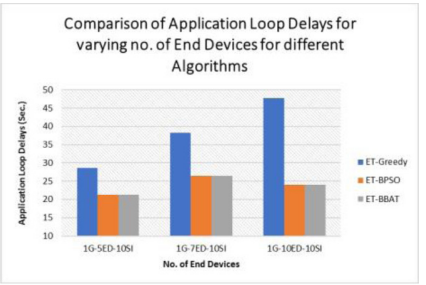

Figure 5

Application Loop Delays (ALD) for simulated algorithms: Greedy, BPSO, and BBAT

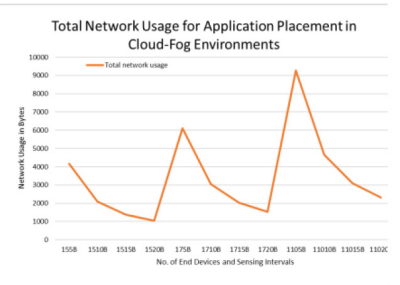

Figure 6

Network usage vs Number of Devices and Sensing Interval

given configurations, it suggests that the Greedy algorithm may not be suitable or effective for solving the placement problem in the context of cloud-fog computing with the provided constraints and settings.In such cases, it becomes important to investigate why the Greedy algorithm failed to find a valid placement. Possible reasons include the algorithm's limitations in handling complex placement scenarios, insufficient exploration of the solution space, or the algorithm being sensitive to the specific problem instance.To address this issue, alternative algorithms such as BPSO and BBAT have been used, as they may offer better exploration and exploitation capabilities to find feasible placements.

The Figure 5 show the graph of Application Loop Delays (ALD) for simulated algorithms such as Greedy, BPSO, and BBAT while Network usage vs Number of Devices and Sensing Interval graph is given in Figure 6.

The analysis reveals insights as follows: ALD-Greedy shows the highest latency per module, indicating suboptimal placements. ALD-BPSO and ALD-BBAT have consistently lower average latencies per module. In IFogSim, total network usage is the aggregated data transmitted during simulation, depicting communication between entities like VMs, End devices, and Gateways. Analyzing BBAT results in cloud fog computing, trends emerge: (1) Longer Sensing Intervals (SI) reduce network usage due to decreased communication frequency between fog nodes. (2) Increasing End Devices (ED) raises network usage, as more data transmission occurs. Overall, network usage is influenced by end devices, sensing intervals, and scenario configurations.

6. Conclusions

The analysis of execution time and loop delays for Greedy, BPSO, and BBAT algorithms in cloud-fog computing reveals insights into their performance. BBAT consistently outperforms Greedy and BPSO in execution time, indicating its efficiency in optimal or near-optimal placements. Greedy struggles with complex scenarios. BBAT's analysis of total network usage indicates lower usage with longer sensing intervals but higher usage with more end devices. This study opens avenues for future exploration: Real-world Deployment, Dynamic Placement, Energy-aware and Secure Placement, and Multi-objective Optimization.

References

[1] Atitallah, S. B., Driss, M., Boulila, W., & Ghézala, H. B. : Leveraging Deep Learning and IoT big data analytics to support the smart cities development: Review and future directions. *Computer Science Review,* 36, 100276 (2020). doi:10.1016/j.cosrev.2020.100276.

[2] Palattella, M. R., Dohler, M., Grieco, L. A., et al. : Internet of things in the 5G era: Enablers, architecture, and business models. *IEEE Journal on Selected Areas in Communications*, 34(3), 510-527 (2016).

[3] Lu, C., Sridharan, A., Krishnamachari, B., & Abdelzaher, T. : Efficient fog data analytics: A distributed edge machine learning approach. In Proceedings of the 19th International Middleware Conference, pp. 81-93 (2018).

[4] Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. : A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials, 19(4), 2322-2358 (2017).

[5] Bonomi, F., Milito, R., Natarajan, P., & Zhu, J. : Fog computing: A platform for internet of things and analytics. In Bessis, N., & Dobre, C. (Eds.), Big Data and Internet of Things: A Roadmap for Smart Environments, pp. 169-186 (2014). Springer International Publishing.

[6] Almazaydeh, S., Hassanat, A. B., Al-Momani, M. A., & Rawashdeh, A. : Bat algorithm for task scheduling in fog computing environment. Future Generation Computer Systems, 97, 367-380 (2019). doi:10.1016/j.future.2018.12.034.

[7] Farhat, H., Al-Fayoumi, M. A., Aljarah, I., & Mirjalili, S. : A hybrid ant colony optimization and artificial bee colony algorithm for task

scheduling in cloud-fog computing. *Future Generation Computer Systems*, 102, 1-15 (2020). doi:10.1016/j.future.2019.07.018.

[8] Liu, N., Wang, Z., Peng, T., & He, X. : A task scheduling algorithm based on bat algorithm and improved ant colony optimization for cloud-fog computing. IEEE Access, 7, 69335-69345 (2019). doi:10.1109/ACCESS.2019.2919124

[9] Abdul-Jabbar, T. A., Hussain, M., & Li, X. : Task scheduling in fog computing environments using the firefly algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 10(6), 2235-2246 (2019). doi:10.1007/s12652-018-0771-1.

[10] Duong, N. H. N., Nguyen, D. T., & Nguyen, D. C. : Metaheuristic approaches for task scheduling in cloud computing environments: A comprehensive survey. *Computers & Operations Research*, 109, 211-236 (2019). doi:10.1016/j.cor.2019.05.014

[11] Wu, Y., Zhang, Y., & Zhang, Y. : Task scheduling algorithm in cloud-fog computing based on improved ant colony optimization. *The Journal of Supercomputing*, 75(2), 779-800 (2019). doi:10.1007/s11227-018-2688-x.

[12] Patil, S. S., Sahoo, S., & Panda, S. : Whale optimization algorithm-based task scheduling in cloud-fog computing environments. *Cluster Computing*, 23(1), 45-59 (2020). doi:10.1007/s10586-019-02942-3.

[13] Rajput, R. S., Singh, N. K., & Singh, V. P. : Task scheduling in cloud-fog computing using grey wolf optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 11(2), 647-657 (2020). doi:10.1007/s12652-019-01588-z.

[14] Zhao, Y., Guo, S., Wang, Y., & Zhang, X. : A hybrid bee colony optimization algorithm for task scheduling in cloud-fog computing. *Journal of Ambient Intelligence and Humanized Computing*, 10(9), 3357-3367 (2019).

[15] Yang, S. S., Wang, L., & Li, Z. M. : A modified imperialist competitive algorithm for task scheduling in cloud-fog computing. *The Journal of Supercomputing*, 76(1), 92-107 (2020).

[16] Khan, M. H., Javaid, N., Javaid, A., & Qasim, U. : Task scheduling in cloud-fog computing using a hybrid algorithm based on grey wolf optimizer and particle swarm optimization. *Journal of Ambient Intelligence and Humanized Computing*, 10(9), 3389-3399 (2019).

[17] Zhang, Z., Liu, H., & Zhou, Y. : An energy-aware task scheduling algorithm for IoT-enabled cloud-fog computing. *Journal of Ambient Intelligence and Humanized Computing*, 10(5), 1775-1786 (2019).

[18] Jiang, J., Wan, J., & Zhu, R. : Efficient task scheduling for fog-enabled IoT systems: A multi-objective optimization approach. *IEEE Transactions on Industrial Informatics*, 14(7), 3048-3056 (2018).

[19] Tanwar, S., Kumar, S., & Kumar, N. : Optimal resource provisioning for task scheduling in edge computing: A multi-objective approach. Future Generation Computer Systems, 89, 267-277 (2018).

[20] Bilal, M., Xu, Y., & Xu, X. : A multi-objective task scheduling algorithm for cloud-fog computing systems. *Journal of Supercomputing,* 77(4), 3592-3607 (2021).

[21] Zhang, W., Zhang, Y., & Wang, L. : Dynamic resource allocation in fog computing: A reinforcement learning approach. *IEEE Internet of Things Journal*, 7(4), 3114-3124 (2019).

[22] Wang, Z., Zhang, X., & Feng, Q. : An improved multi-objective evolutionary algorithm for task scheduling in cloud-fog computing. IEEE Access, 8, 147255-147267 (2020).

[23] Juo, L., Yin, L., Hu, J., Wang, C., Liu, X., Fan, X., ... & Liu, L. : Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT. *Future Generation Computer Systems,* 100, 796-807 (2019).

[24] Smeto: Stable matching for energy-minimized task offloading in cloud-fog networks. IEEE 90th Vehicular Technology Conference, VTC-Fall (2019).

[25] Nasir, M. H., Khan, S. A., Khan, M. M., & Fatima, M. : Swarm intelligence inspired intrusion detection systems—a systematic literature review. Computer Networks (2022).

Books:

[26] Mahmud, R., & Buyya, R. : Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit. In Buyya, R., & Srirama, S. (Eds.), Fog and Edge Computing: Principles and Paradigms (Chapter 17). Wiley STM (2018).