Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 46 (2025), No. 6, pp. 1871–1879
DOI : 10.47974/JIOS-2016

Energy-efficient resource allocation in fog computing using hybrid genetic algorithm-based VM consolidation

Manjula Gururaj Rao *
Department of Information Science and Engineering
Nitte Mahalinga Adyantaya Memorial Institute of Technology
NITTE (Deemed to be University)
NITTE Karkala
Udupi 574110
Karnataka
India

H. Priyanka †
Department of Computer Science & Engineering
PES University
Bengaluru 560085
Karnataka
India

H. Gururaj Rao §
Department of Talent Aquistion Team-Campus Connect
Sasken Technologies Ltd
Bangalore 560071
Karnataka
India

Abstract

Fog computing brings cloud-like services closer to data sources, improving responsiveness but also introducing challenges like high energy consumption and inefficient resource use. To tackle this, VM consolidation using live migration is employed

* *ORCID-ID:* **0000-0003-0957-5786**
† *ORCID-ID:* **0000-0002-4155-0418**
§ *ORCID-ID:* **0009-0009-9814-2704**

* *E-mail:* **dr.manjulagururajh@gmail.com (Corresponding Author)**
† *E-mail:* **priyankahsachin@gmail.com**
§ *E-mail:* **gururajraoh@gmail.com**

to enhance energy efficiency and resource management. This paper introduces a hybrid Genetic Algorithm model that combines various selection strategies—Tournament, Rank, SUS, Truncation, and Roulette-Wheel—to optimize CPU and memory usage during VM consolidation. By using heuristics for population generation, fitness evaluation, and load balancing, the model minimizes active physical machines and adapts to workload changes, improving efficiency in fog data centers.

Subject Classification: *68M20, 68U20, 68T20.*

Keywords: *SUS, GA, VM, PM, hybrid GA, Memory utilization, CPU utilization.*

1. Introduction

The evolution of Internet of Things (IoT) and the growing demand for real-time data processing have highlighted the shortcomings of centralized cloud systems. In response, fog computing has emerged as a viable solution by extending cloud-like capabilities closer to data sources. By providing computation, storage, and networking near the edge, fog computing enhances responsiveness and supports latency-sensitive applications like intelligent transport systems and video conferencing. It operates through a decentralized framework where edge devices communicate with nearby fog nodes, such as routers and gateways, which are capable of handling localized tasks. This local processing reduces pressure on central servers, minimizes latency, and improves decision-making in time-critical scenarios. Alongside fog nodes, more powerful fog servers are deployed in locations like shopping malls to manage virtualized tasks such as job scheduling and intermediate storage. The architecture facilitates layered data handling by first processing data at the edge and then forwarding complex tasks to fog servers or the cloud. With the use of technologies like Wi-Fi, Ethernet, and 5G, fog computing ensures seamless communication, efficient resource use, and reduced reliance on centralized systems. While it improves performance, the rapid growth of fog infrastructure raises concerns about energy consumption and inefficient resource utilization. To address this, research proposes hybrid optimization techniques using genetic algorithms to improve VM consolidation, job scheduling, and energy efficiency, thus contributing to a more sustainable and effective fog computing model.

2. Literature Review

Cloud computing data centers are rapidly expanding to meet high-performance demands, yet inefficient resource utilization results in high energy consumption and costs, leading researchers like [1] to propose VM consolidation techniques that factor in multiple efficiency parameters. Building on this, genetic algorithms have gained popularity for optimization in fog environments, with [2] offering a detailed taxonomy and [3] integrating GA with reinforcement learning to improve energy efficiency in fog nodes. Although fog computing effectively supports latency-sensitive applications, it still struggles

with load balancing and resource allocation, [4] propose a two-layer GA-based VM consolidation architecture, while [5, 6] present advanced hybrid optimization strategies like MOD-JAYA, E2FFO, and DCCO to boost energy efficiency and system performance. Author in [7] propose the HEPGA algorithm by combining HEFT, PSO, and GA for optimal task scheduling, while [8] apply a whale optimization hybrid to address cost, SLA violations, and resource efficiency. Building on this, researchers like [9, 10] have leveraged advanced meta-heuristics for load balancing and resource scheduling, achieving reductions in energy consumption and improvements in QoS. Innovations such as MOMA [11], DPSO-GA with CNN-LSTM [12], and WOA-Scheduler [13] have further advanced workload provisioning and dynamic task scheduling, with MG RAO [14] exploring resource utilization in WSN-IoT contexts using varied algorithms. Additionally, [6] introduce energy-efficient scheduling models like GA-based ILP, while [1] focus on workload learning and VM migration, all contributing to scalable and sustainable fog and cloud computing through AI and hybrid algorithm integration.

3. Proposed Methodology

This paper presents a VM optimizer integrated within the Fog Node controller module, designed to enhance task scheduling and resource allocation in fog computing environments. The Fog controller comprises three primary units: the VM Controller, GA Optimizer, and Computing Unit, each playing a vital role in managing virtual machines efficiently as depicted in the Figure 1. The VM Controller is responsible for monitoring VM status, analyzing application requirements, and prioritizing service requests based on urgency and resource needs. It collects data on context instances and system status to queue and assign service requests accordingly. The GA Optimizer employs a Hybrid Genetic Algorithm, including a variant with Local Search (HGALS), to select VMs based on CPU and memory utilization. Various fitness functions were

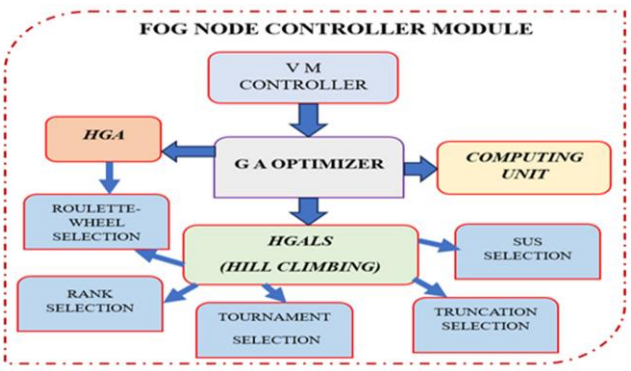

Figure 1
Proposed Flow diagram the Fog Node

tested to improve the accuracy of VM selection and optimize resource usage. Once a VM is selected, it is passed to the Computing Unit, which represents virtualized computing resources responsible for executing delay-sensitive tasks. This Computing Unit helps reduce latency by processing data locally within the fog environment. The use of hybrid optimization techniques ensures better performance, especially for applications requiring real-time processing.

2.1 *Algorithms*

Algorithm 1: GA with the Roulette-Wheel Selection

Input : T_R- Task Resources (T_{RCPU}- CPU and T_{RM}- Memory)
P_{size}- Population Size, G-Generation, M_R- Mutation Rate; T_{COUNT} – Task Count; P_{NEW}- New Population

Algorithm Genetic_Algorithm(T_R, P_{size}, G, M_R):

1. Initialize population using Initialize_Population (T_{COUNT}, P_{size})

$$P = \{i = 1,2, \dots \dots . P_{size}\}$$

$$chromosome = \{j = 1,2, \dots \dots T_{COUNT}\}$$

2. For generation in range (1, G + 1):
   a. Evaluate fitness for each chromosome in the P
      For each chromosome:
      Compute total CPU and memory usage is given by equation 1 and fitness given by equation 2.

$$Penalty = Max(0, T_{RCPU} - CPU\_LIMIT) + Max(0, T_{RM} - MEMORY\_LIMIT) \quad (1)$$

$$FITNESS = \frac{1}{(1+Penalty)} \quad (2)$$

   b. Create new population:
      For i = 1 to P_{size}/ 2:
      i. Select parents using Selection (P_{size}, fitness_scores). The selection is based on the Roulette- Wheel. The parents are selected based on the equation 3.

$$P(chromosome_i = \frac{Fitness(chromosome_i)}{\Sigma_{chromosome} \quad Fitness}) \quad (3)$$

      ii. Perform crossover to generate child_1, child_2.

$$child\_1 = parent\_1[:point] + parent\_2[:point]$$
$$child\_2 = parent\_2[:point] + parent\_1[:point]$$

      iii. Mutate child_1 and child_2 with Mutate().
      iv. Add child1 and child2 to P_{NEW}
   c. Replace the old population with the P_{NEW} using the crossover and mutate.
   d. Print the best fitness of the generation.
   e. If best fitness == 1:
      Break (optimal solution found).
3. Return the best chromosome as the optimal schedule.

End Algorithm

HGA can improve resource utilization and reduce energy consumption in fog computing by combining GA with heuristics or other optimization techniques. To further enhance performance, HGA is modified with a Local Search using hill-climbing, forming the HGALS. The paper also introduces features to track and visualize CPU and memory usage across generations during HGALS execution.

Algorithm 2: HG with Local Search, Roulette-Wheel Selection

Input: P_{size}- Population Size, G-Generation, M_R- Mutation Rate; T_{COUNT} – Task Count; P_{NEW}- New Population

T- Tasks$\{i = 1,2,3 \ldots\ldots\}$; PM- Physical Machine contains the information regarding the CPU and memory

Define Hybrid_Genetic_Algorithm(T, PM, P_{size}, generations, M_R):

T_{COUNT} = len(tasks)

population = Initialize population with random task assignments

best_fitness_per_gen=[],cpu_utilization_per_gen=[],

memory_utilization_per_gen = []

For gen from 0 to generations - 1:

Fitness_scores = [fitness(individual, tasks, PM) for individual in population]

new_population = []

For each pair of parents:

Parent_1, parent_2 = selection(population, fitness_scores)

Child_1, child_2 = crossover(parent1, parent2)

new_population.extend([mutate(child1, mutation_rate, task_count), mutate(child_2, mutation_rate, task_count)])

top_individuals = sort population by fitness in descending order

optimized_individuals = Apply local search to top 5 individuals

new_population [0 to len (optimized_individuals)] = optimized_individuals

population = new_population

best_fitness = max ([fitness (ind, tasks, PM) for ind in population])

best_fitness_per_gen. append (best_fitness)

best_individual= population [fitness_scores. index (max(fitness_scores))]

cpu_util, memory_util = calculate_utilization (best_individual, tasks, PM)

cpu_utilization_per_gen. append (cpu_util)

memory_utilization_per_gen. append (memory_util)

Return population, best_fitness_per_gen, cpu_utilization_per_gen, memory_utilization_per_gen

The proposed approach optimizes task scheduling in fog computing by integrating a GA with Local Search, considering CPU and memory constraints. It begins with a random population of task-to-PM schedules, evaluated using a fitness function that penalizes high resource usage. Through crossover,

mutation, and roulette-wheel selection, new populations are generated while maintaining diversity. The best individuals undergo hill-climbing-based local search to enhance performance. Progress is monitored by recording fitness values and resource utilization across generations. HGALS is tested using five different fitness functions: Rank Selection (RS), Stochastic Universal Sampling (SUS), Tournament Selection (TS), Truncation Selection, and Roulette-Wheel Selection (RW). Each technique offers unique benefits in balancing selection pressure and diversity, contributing to better scheduling efficiency and energy savings in the fog environment.

2.2 *Experimental Analysis & Result*

In the experimental results Initial step, the CPU utilization is checked. The proposed approach compares different selection methods—RW, RS, SUS, TS, and Truncation Selection, applied to both HGA and HGALS. In RW, individuals are chosen probabilistically based on fitness, maintaining diversity but risking premature convergence if fitness variance is high or low. RS improves fairness by ranking individuals before assigning selection probability, enhancing performance while preserving diversity. SUS selects parents using evenly spaced pointers, reducing randomness and maintaining proportional representation, which improves scheduling consistency. TS boosts selection pressure by choosing the best among randomly sampled groups, ensuring robust selection while maintaining fairness. Truncation Selection quickly retains only the top-performing individuals but risks reduced diversity and premature convergence, though it is effective in simpler, well-understood environments.

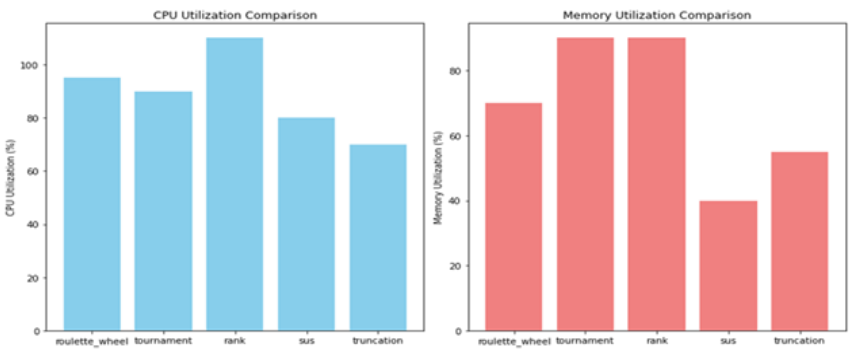

Figure 2
CPU and Memory utilization of the HGALS

All methods are evaluated for CPU and memory utilization, demonstrating the impact of each selection approach on resource efficiency. The combined results highlight that the choice of selection function significantly influences the optimization quality and system performance in fog computing environments. Table 1 presents the CPU and memory utilization, while Figure 2 visually represents the same.

Table 1

HGALS with the Different Selection methods

	Selection Method	Best Fitness	CPU Utilization (%)	Memory Utilizatio
0	roulette_wheel	0	95.0	70.0
1	tournament	0	90.0	90.0
2	rank	0	110.0	90.0
3	sus	0	80.0	40.0
4	truncation	0	70.0	55.0

Figures 3 and 4 illustrate the selection of best fitness and VM-PM utilization, where for 50 instances, 10 VMs are efficiently mapped to 4 PMs based on their best-fit assignment.

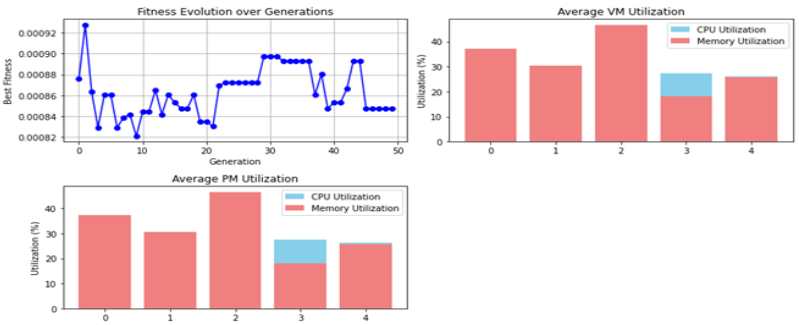

Figure 3

VM selection and assignment of VM to the PM.

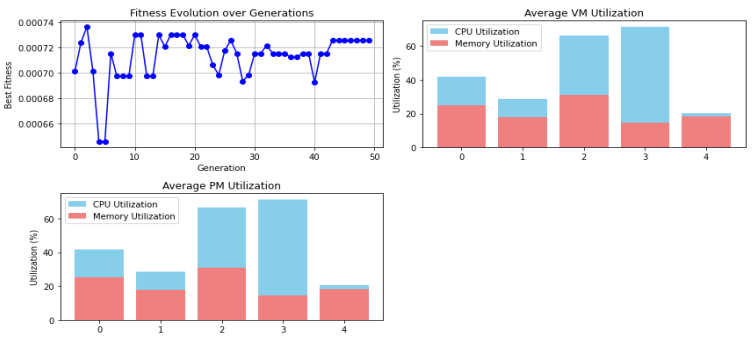

Figure 4

Selection-utilization-Best fitness and VM and PM

4. Conclusion

Fog computing, offering low-latency benefits for delay-sensitive applications, faces high energy and operational costs due to inefficient resource management in expanding data centers. This paper explores a HGA and HGALS

model to address resource utilization and energy efficiency challenges in fog computing environments. The proposed model employs selection strategies like Tournament, Rank, SUS, Truncation, and Roulette-Wheel to optimize CPU and memory utilization during VM consolidation. By incorporating heuristic-based initialization and local search strategies like hill-climbing, it dynamically adapts to workload changes, enhancing energy savings and system efficiency. Performance analysis demonstrates the model's superiority in resource optimization, ensuring quality of service and effective scheduling for diverse tasks. The hybrid GA approach is a scalable, energy-efficient solution, with potential for future refinement using advanced metaheuristic methods and real-world applications.

References

[1] T. K. Mohammed, D. N. Vasundhara, S. H. Mehanoor, E. Sreedevi, P. R. Kumar, C. Manihass, and S. F. Baba, "A novel fusion approach for advancement in crime prediction and forecasting using hybridization of ARIMA and recurrent neural networks," *Journal of Information Systems Engineering and Management*, vol. 10, no. 38, pp. 404–420 (2025).

[2] I. S. Ahmed, H. F. Abbas, and S. H. Asaad, "Study the new generalization of fuzzy s-algebra namely fuzzy P*-algebra," *J. Discrete Math. Sci. Cryptogr.,* vol. 28, no. 4-A, pp. 1097–1103 (2025).

[3] R. K. Moje, B. Tiple, S. M. Patil, T. Jadhav, A. P. Munshi, and A. Revekar, "A framework for multi-task learning optimization in deep neural networks: Balancing task priorities for improved performance," *Journal of Information and Optimization Sciences*, vol. 46, no. 4-B, pp. 1141–1151 (2025).

[4] G. B. Mohammad, S. Shitharth, and P. R. Kumar, "Integrated machine learning model for an URL phishing detection," *Int. J. Grid Distrib. Comput.,* vol. 14, no. 1, pp. 513–529 (2021).

[5] B. Shilpa, P. R. Kumar, and R. K. Jha, "LoRa DL: A deep learning model for enhancing the data transmission over LoRa using autoencoder," *The Journal of Supercomputing*, vol. 79, pp. 17079–17097 (2023).

[6] S. Gautam, S. S. Gaur, and S. Masood, "A comparative study of certificate and certificate-less cryptographic algorithm and its energy consumption analysis in WSN," *TARU J. Comput. Dyn. Technol. Adv.,* vol. 1, no. 2, pp. 85–100 (2019).

[7] M. Hanif, U. Shahzad, I. Shahzadi, and N. Koyuncu, "Stochastically increasing grouped data using the MLE of mean of the generalized exponential distribution," *TARU J. Organ. Behav. Analytics,* vol. 1, no. 1, pp. 65–82 (2019).

[8] B. Shilpa, P. R. Kumar, and R. K. Jha, "Spreading factor optimization for interference mitigation in dense indoor LoRa networks," in Proc. 2023 IEEE IAS Glob. Conf. Emerg. Technol. (GlobConET), pp. 1–5 (May 2023).

[9] N. Arivazhagan, K. Somasundaram, D. Vijendra Babu, M. Gomathy Nayagam, R. M. Bommi, G. B. Mohammad, … V. Prabhu Sundramurthy, "Cloud-Internet of Health Things (IOHT) Task Scheduling Using Hybrid Moth Flame Optimization with Deep Neural Network Algorithm for E Healthcare Systems," Scientific Programming, vol. 2022, no. 1, pp. 4100352 (2022).

[10] M. N. Nikhate, K. L. Bondar, and S. B. Kiwne, "Fixed point theorems for self-mappings in generalized metric spaces," *Journal of Dynamical Systems and Geometric Theories*, vol. 21, no. 2, pp. 121–126 (2023).

[11] R. Ranjan, D. Pandey, A. K. Rai, D. Gupta, P. Singh, P. R. Kumar, and S. N. Mohanty, "A manifold-level hybrid deep learning approach for sentiment classification using an autoregressive model," *Applied Sciences,* vol. 13, no. 5, pp. 3091 (2023).

[12] P. Li, H. Wang, G. Tian, and Z. Fan, "Towards Sustainable Cloud Computing: Load Balancing with Nature-Inspired Meta-Heuristic Algorithms," *Electronics*, vol. 13, no. 13, pp. 2578 (2024).

[13] S. M. Kak, P. Agarwal, M. A. Alam, and F. Siddiqui, "A hybridized approach for minimizing energy in cloud computing," *Cluster Comput.,* vol. 27, no. 1, pp. 53–70 (2024).

[14] Z. S. Alsham, E. Bahçekapılı, and A. Ayaz, "Trends in IoT applications in smart campuses: A topic modeling approach," *COLLNET Journal of Scientometrics and Information Management*, vol. 19, no. 1, pp. 21–40 (2025).

Received December, 2024