

Random Forest Based Edge Load Balancing of IOT Devices

Swati Jat¹, Assistant Professor Rani Kushwaha², Professor Jayshree Boaddh³

M.Tech student, Mittal Institute of Technology, Bhopal¹

CSE Mittal Institute of Technology, Bhopal²

HOD, Cse Vaishnavi Institute of technology and Science, Bhopal³

Abstract- IoT device-based communication boosts monitoring, business operations, and daily activities but also increases the load on servers and clouds. To handle this, edge computing acts as an intermediary layer. Efficient job management is critical for large-scale IoT networks, but existing models often fail to adapt based on past job sequences. This work introduces a model using a modified wolf Optimization algorithm to dynamically balance loads without prior training. It also incorporates a Random Forest model to generate initial job sequences. Experiments show that the proposed approach reduces job makespan time and enhances edge resource utilization compared to other models.

Index Terms- Job edge load, Cloud Computing, Genetic Algorithm.

I. INTRODUCTION

The rapid and pervasive integration of low-cost, miniature network devices into the Internet has led to an extraordinary surge in connectivity. Moving beyond traditional computing machines, billions of smart 'things' now independently interact and communicate, collectively forming the Internet of Things (IoT) [1, 2]. This network of smart objects extends to constrained devices such as low-power wireless sensor nodes, which are defined by their limited processing power and memory capabilities. Despite these constraints, IoT environments allow end-users to seamlessly access resources, services, and capabilities, often without an awareness of the underlying infrastructure. Users benefit from this integration as smart devices create a cohesive, connected environment that streamlines daily activities and delivers intelligent services anytime and anywhere.

Within this expansive IoT framework, low-power wireless sensor devices are crucial for collecting and transmitting essential physical data, such as temperature, pressure, and motion. This data serves as the backbone for intelligent service applications, which span from environmental monitoring to real-time decision-making for optimized management processes.

A key component in the efficient operation of IoT networks is load balancing (LB), which is responsible for allocating resources to user tasks in an optimal manner. This allocation enhances overall resource utilization within the network [3]. Resources in IoT networks include the physical and computational capabilities of nodes, such as processing power, memory storage, and energy, alongside network resources like

bandwidth, load balancers, and traffic analyzers. By implementing effective LB techniques, IoT networks can prevent system overloads and improve overall performance, thus enhancing Quality of Service (QoS) metrics, such as response times, throughput, and resource utilization [4].

The vast number of events generated by IoT devices can lead to substantial traffic on specific data paths, causing network congestion and degraded performance. When traffic is not distributed evenly across network paths, latency increases, packets may be lost, and the Packet Delivery Ratio (PDR) suffers.

Efficient load balancing methods are, therefore, necessary to avoid congestion. Such methods utilize local network information, including network topology, to distribute workloads evenly among various routes and resources, ultimately enhancing network performance.

Parallel to the growth of IoT, the concept of edge cloud computing has gained prominence. This paradigm supports real-time services and offers rapid responsiveness for end-users by delivering partial cloud computing services closer to where data is generated—at the network's edge.

Edge computing reduces latency and improves user experiences [6, 7]. However, the limited capabilities and resources of edge devices, compared to centralized cloud servers, present challenges for load balancing. Thus, tailored control mechanisms within edge cloud computing environments are essential to maximize resource utilization and ensure optimal performance of the network [8].

II. RELATED WORK

Hu, P. et al. (2019) highlighted in their research [9] that while examining global popularity can offer valuable insights into common content request trends among users, it fails to accurately capture individual user preferences. They argue that a more effective strategy for maximizing the click-through rate would involve focusing on individual user preferences rather than relying solely on broad, global trends. To achieve a more accurate and relevant load prediction, their study proposes incorporating personalized load prediction for each user.

In their work, Kong, W. et al. (2020) explored AI-driven IoT applications, such as video processing at the edge [10]. Unlike prior studies, their approach addresses the diversity present in serverless edge systems, including variations in hardware, platforms, and software layers. Their goal is to create load-balancing policies that go beyond latency-awareness, taking into consideration multiple performance metrics such as cost efficiency, throughput, energy consumption, and AI inference accuracy.

Saba, T. et al. (2021) developed a secure data management framework featuring a distributed load-balancing protocol that employs particle swarm optimization [12]. Their aim was to reduce response times for cloud service users while maintaining secure network communications. By utilizing distributed computing strategies and bringing complex computations closer to the requesting nodes, the proposed system seeks to minimize latency and reduce transmission overhead. Moreover, it evaluates trust levels in a controlled manner to protect communication channels from potentially malicious devices.

S. Shao et al. (2022) presented a load-balancing algorithm for edge computing called LBA-EC, which utilizes a weighted bipartite graph [13]. This algorithm focuses on optimizing network edge resource utilization to decrease user delays and improve service delivery. Their task scheduling process is executed in two phases: first, tasks are matched to various edge servers, and in the second phase, tasks are allocated to containers within these servers based on parameters like energy consumption and task completion time.

In a 2023 study, Y. Wang et al. introduced an edge-computing load-balancing technique specifically designed for Low Earth Orbit (LEO) satellite networks [14]. This method maximizes the flow of virtual links by identifying minimum computing node rectangles, establishing virtual edge computing links between nodes and users, and employing the Ford-Fulkerson algorithm to achieve maximum network flow. The method supports efficient resource allocation for computing and transmission operations.

Research by P. V. Lahande et al. (2024) concentrated on load-balancing mechanisms within the WorkflowSim environment using the Sipt task dataset [15]. They tested various load-balancing algorithms, including First Come First Serve (FCFS), Maximum-Minimum (Max-Min), Minimum Completion Time (MCT), Minimum-Minimum (Min-Min), and Round-Robin (RR). Their study involved four different phases with variations in task lengths and consisted of sixteen scenarios, each using different numbers of virtual machines (VMs).

III. PROPOSED METHODOLOGY

In this section, the proposed Random Forest & wolf based Edge Load Balancing (RFW-ELB) method is thoroughly described. The approach begins with the initialization of each edge node within the network, where each node is allocated specific resources. This initial allocation process, as defined by the proposed model, is illustrated in Figure 1. Each component shown in Figure 1 is explained in detail within this section, while relevant notations are summarized in Table 1. The dynamic sequencing of IoT jobs is managed using the Artificial Immune Genetic Algorithm. This algorithm not only handles job sequences and available resources but also facilitates the generation of an initial population. The Random Forest model plays a key role in this learning process.

Edge Resources: In this context, each edge device within the network possesses a unique set of resources, which include R_m (Edge Resource Memory), R_p (Edge Resource Processor), and R_b (Edge Resource Bandwidth). This study assumes that the network contains a total of n_e edge nodes, which are responsible for managing N_j IoT jobs [16]. The proposed model aims to determine an IoT job sequence that minimizes the makespan time, thereby optimizing task completion efficiency within the network.

Random Forest Learning: To enhance the efficiency of IoT job scheduling, a variety of job requirements are generated for IoT devices, and corresponding job sequences for edge nodes are determined. A Random Forest model [17] is trained to create the optimal job sequence for IoT devices. Once trained, the model serves as the basis for the initial population generation within the genetic algorithm. Unlike many genetic approaches that rely on Gaussian functions for population generation, this method leverages learned job sequences to identify patterns in task allocation. The training vector for the Random Forest model comprises IoT resource information, including processor capability, memory capacity, and bandwidth. In addition to these parameters, the vector also includes the average available resources that can be allocated by the edge nodes. By combining these two types of features, the Random Forest model gains a comprehensive understanding of the resource allocation landscape and effectively supports the proposed edge load-balancing approach.

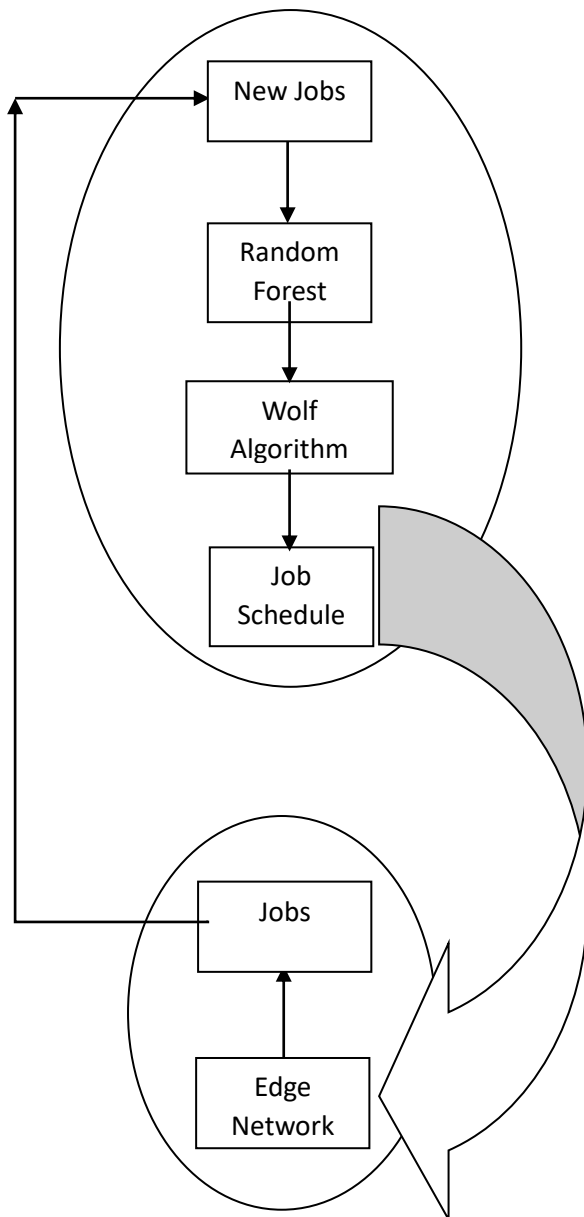


Fig.1 Proposed Edge load balancing model

Wolf Population:

Wolfs are group of job sequence. So a wolf is a vector of j number of jobs. So if w number of wolfs generate then JC is wolf population matrix having wxn dimension. Selection of n number of job sequence in vector was generate by random function.

$$JC \leftarrow \text{Generate_Herd}(\text{Random_Forest}, \text{Jobs}, j, n)$$

Fitness: Each wolf in the population is evaluated to determine its suitability for the given problem. This function computes a single value for each wolf, representing its fitness score. Equation 1 defines the fitness function for this task, calculating the fitness value F.

$$F_{JC} = \text{Makespan}(JC)$$

Update Wolf position

Once F_{JC} value obtain by fitness function then sort F_{JC} in descending order and find wolf sub category. First sorted fitness value is consider as alpha wolf, then next $m/3$ consider as beta wolfs and next $m/3$ wolfs consider as delta wolfs. Position of each wolf were modified by the Eq 2 to 4, [15].

$$A = \text{Pos} - Ct * (\text{Pos}/Mt) * r - \text{Pos} - Ct * (\text{Pos}/Mt) \text{---Eq. 2}$$

$$D_w = c * (\text{Delta_Wolf} - \text{Alpha_Wolf}) \text{---Eq. 3}$$

$$X_1 = \text{Delta_Wolf} - A * \text{abs}(D_w) \text{---Eq. 4}$$

Where Pos is position of wolf range {0,1,2,3} from Prey, c obtain by $\text{Pos} * r$ and r is random number range from {1 to n}.

Similarly

$$D_w = c * (\text{Beta_Wolf} - \text{Alpha_Wolf}) \text{---Eq.5}$$

$$X_2 = \text{Beta_Wolf} - A * \text{abs}(D) \text{---Eq. 6}$$

$$X_3 = \text{Alpha_Wolf} \text{---Eq. 7}$$

Final position shifting value estimate by Eq. 8

$$X = \frac{X_1 + X_2 + X_3}{3} \text{---Eq. 8}$$

Update: The best solution, referred to as the alpha wolf, is determined based on the fitness scores of each wolf in the population's clan [18]. Several status parameters are randomly modified based on the characteristics of the best wolf. Cloning involves replicating the top-performing wolf job sequence to other wolf at X positions.

$$JC \leftarrow \text{populaiton_update}(Mb, JC, X)$$

Hunting Rule

After t number of iteration steps (fitness function, wolf position update, crossover) final wolf population pas through fitness function and best fitted wolf is consider as alpha wolf. This wolf element pages are predicted possible set of job sequence.

IV. EVALUATION PARAMETERS

Experiment work was done on different environmental conditions, where number of edges, and IOT job sequence get modified. Implementation of model was done on MATLAB software and machine having a configuration of processor I312th generation with 8GB RAM. Comparison of Random Forest & wolf based Edge Load Balancing (RFW-ELB) model was done by existing Preference Based Stable Matching model (PBSM) proposed in [20].

Results

Table 1 Makespan based comparison of edge load balancing models.

Edges	IOT Jobs	RFW-ELB	PBSM
30	100	166.79	166.9154
60	100	165.8886	168.3107
90	100	181.0412	182.3779
120	40	136.7515	140.4765
120	60	173.53	174.7867
120	80	158.127	159.0196
120	100	178.6535	178.1476

Table 1 shows that proposed RFW-ELB has reduces the makespan time of the edge jobs. In all experimental setup of different number of jobs and edges proposed model takes less time. Use of random forest for population generation has increases the prediction of job set. Table 1 shows that makespan was reduces by 0.79% as compared to existing model.

Table 2 Total Flow Time based comparison of edge load balancing models.

Edges	IOT Jobs	RFW-ELB	PBSM
30	100	16680	16692
60	100	16589	16831
90	100	17923	18238
120	40	5333.3	5619.1
120	60	10238	1048.7
120	80	12492	12722
120	100	17865	17815

Edge load balancing total flow time shown in table 2. It was found that use of wolf for job sequencing has reduces the flow time of model.

Table 3 Edge Utilization based comparison of edge load balancing models.

Edges	IOT Jobs	RFW-ELB	PBSM
30	100	84.9667	77.9667
60	100	71.9833	59.9833
90	100	54.5342	42.9889
120	40	194.8718	170
120	60	191.5254	165
120	80	24.0401	8.7396
120	100	37.9917	27.9917

Table 3 shows that proposed RFW-ELB has increases the edge utilization of the system. In all experimental setup of different number of jobs and edges proposed model increases the utilization of nodes. Use of wolf for job set prediction has increases the system utilization. Table 3 shows that edge utilization was increased by 16.25% as compared to existing model.

Table 4 Execution time based comparison of edge load balancing models.

Edges	IOT Jobs	RFW-ELB	PBSM
30	100	0.044	0.0435
60	100	0.037	0.0392
90	100	0.0334	0.036
120	40	0.0199	0.0303
120	60	0.0221	0.0227
120	80	0.0348	0.0309
120	100	0.0326	0.0339

Execution time need for generating the job sequence show in table 4. It was found that use of random forted and wolf algorithm has reduces the time. As genetic works on randomization so time is optimized to produce new job sequences.

V. CONCLUSION

This paper has developed a model that utilizes the edge resource features for learning the job sequences. Job patterns were used for the training of random forest that will generate job sequences as per current requirement. Wolf optimization algorithm uses random forest for the population generation of job sequences and generate final job. Experiment was done on different set of edge nodes with heterogeneous configuration and different number of jobs. Result shows that proposed has reduces makespan by 0.79% as compared to existing model, further increases edge utilization by 16.25% as compared to existing model. In future scholars can improved same model in IOT networks as well.

REFERENCES

1. Du, M.; Wang, Y.; Ye, K.; Xu, C. Algorithmics of cost-driven computation offloading in the edge-cloud environment. *IEEE Trans. Comput.* 2020.
2. Li, C.; Bai, J.; Chen, Y.; Luo, Y. Resource and replica management strategy for optimizing financial cost and user experience in edge cloud computing system. *Inf. Sci.* 2020, 516, 33–55.
3. Li, C.; Bai, J.; Luo, Y. Efficient resource scaling based on load fluctuation in edge-cloud computing environment. *J. Supercomput.* 2020.

4. Hoang, K.D.; Wayllace, C.; Yeoh, W.; Beal, J.; Dasgupta, S.; Mo, Y.; Paulos, A.; Schewe, J. New distributed constraint reasoning algorithms for load balancing in edge computing. In *International Conference on Principles and Practice of Multi-Agent Systems*; Springer International Publishing: Cham, Switzerland, 2019; pp. 69–86.
5. Liu, Y.; Zeng, Z.; Liu, X.; Zhu, X.; Bhuiyan, M.Z.A. A novel load balancing and low response delay framework for edge-cloud network based on SDN. *IEEE Internet Things J.* 2019.
6. Puthal, D.; Ranjan, R.; Nanda, A.; Nanda, P.; Jayaraman, P.P.; Zomaya, A.Y. Secure authentication and load balancing of distributed edge datacenters. *J. Parallel Distrib. Comput.* 2019, 124, 60–69.
7. Ren, J.; Tian, H.; Lin, Y.; Fan, S.; Nie, G.; Wu, H.; Zhang, F. Incentivized Social-Aware Proactive Device Caching with User Preference Prediction. *IEEE Access* 2019, 7, 136148–136160.
8. Pranveer Singh, Raghuraj Singh Suryavanshi. "Survey on Load Balancing Techniques Shantanu Shukla Research Scholar". *International Journal of Applied Engineering Research* ISSN 0973-4562 Volume 14, Number 2, 2019.
9. Hu, P.; Dhelim, S.; Ning, H.; Qiu, T. Survey on fog computing: Architecture, key technologies, applications and open issues. *J. Netw. Comput. Appl.* 2017, 98, 27–42.
10. Kong, W.; Li, X.; Hou, L.; Li, Y. An efficient and credible multi-source trust fusion mechanism based on time decay for edge computing. *Electronics* 2020.
11. Salehe M., Hu Z., Mortazavi S.H., Capes T., Mohamed I. VideoPipe: Building video stream processing pipelines at the edge *Middleware Industry 2019 - Proceedings of the 2019 20th International Middleware Conference Industrial Track, Part of Middleware 2019 (2019)*, pp. 43-49.
12. Saba, T., Rehman, A., Haseeb, K. et al. Cloud-edge load balancing distributed protocol for IoE services using swarm intelligence. *Cluster Comput* 26, 2921–2931 (2023).
13. S. Shao et al., "LBA-EC: Load Balancing Algorithm Based on Weighted Bipartite Graph for Edge Computing," in *Chinese Journal of Electronics*, vol. 32, no. 2, pp. 313-324, March 2023.
14. Y. Wang et al., "Load-Balancing Method for LEO Satellite Edge-Computing Networks Based on the Maximum Flow of Virtual Links," in *IEEE Access*, vol. 10, pp. 100584-100593, 2022.
15. P. V. Lahande, P. R. Kaveri, J. R. Saini, K. Kotecha and S. Alfarhood, "Reinforcement Learning Approach for Optimizing Cloud Resource Utilization With Load Balancing," in *IEEE Access*, vol. 11, pp. 127567-127577, 2023.
16. Ajay Jangra, Neeraj Mangla. "An efficient load balancing framework for deploying resource scheduling in cloud based communication in healthcare", *Measurement: Sensors*, Volume 25, 2023.
17. Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, 20(1), 3-29.
18. Ali I.M., Sallam K.M., Moustafa N., Chakraborty R., Ryan M., Choo K.-K.R. An automated task scheduling model using non-dominated sorting genetic algorithm II for fog-cloud systems *IEEE Trans. Cloud Comput.*, 10 (4) (2022), pp. 2294-2308.
19. Qiu, Y., Yang, X. & Chen, S. An improved gray wolf optimization algorithm solving to functional optimization and engineering design problems. *Sci Rep* 14, 14190 (2024).
20. A. Bandyopadhyay et al., "EdgeMatch: A Smart Approach for Scheduling IoT-Edge Tasks With Multiple Criteria Using Game Theory," in *IEEE Access*, vol. 12, pp.