Multi-Objective GWO Optimization-based Task Offloading in IoT Network

Suraj Kumar[1] Rakesh Shivhare[2]
[1,2]Department of Computer Science Engineering
[1,2]Radharaman Engineering College, Bhopal (M.P.), India

Abstract — Despite the increasing capabilities of technology, smart and intelligent tasks such as those required for smart healthcare, virtual reality, and IoT-related services are still not possible with current technology. In this work, a multi-objective GWO optimization algorithm was proposed and evaluated for task offloading in fog computing. The results showed improvements in average response time, energy consumption, and load balancing measures, demonstrating the effectiveness of the method. MOGWO was found to be an effective offloading technique for IoT systems that require low latency and quick response times, and can help improve the performance of IoT applications by efficiently utilizing the resources available in the IoT network.

Keywords: Fog Computing, Internet of Things, Task Offloading, Multi-objective.

I. INTRODUCTION

The Internet of Things (IoT) will dominate urban cities and transform overcrowded cities into smart cities. The aim of a smart city is to use cutting-edge IT technology to improve the quality of services offered to residents. To achieve the goal of the smart city, the IoT consists of a large number of geographically distributed nodes. Each node contains a number of smart sensors, where each sensor senses a physical parameter of the surrounding environment within its transmission range. Cloud computing is used as a backend layer for data storage and data analysis because the cloud can provide enormous storage and processing capabilities that are otherwise not available to the IoT devices. This is called task offloading. Fog Computing is the term coined by Cisco that refers to extending cloud computing to an edge of the enterprise's network. Thus, it is also known as Edge Computing or Fogging. It facilitates the operation of computing, storage, and networking services between end devices and computing data centres.

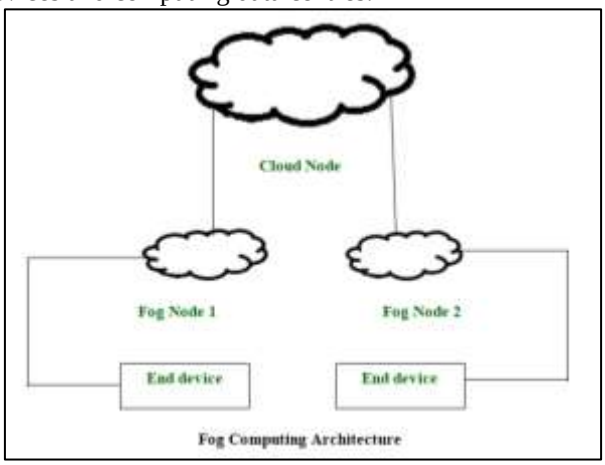

Fig. 1: Fog Computing Architecture

A. Task Offloading in Fog Computing

The task offloading mechanisms are often employed to make room for resources that are used for computation and storage. For instance, it is required to utilize offloading and transfer the data (or tasks) to a new resource when fog node processes do not run properly on their primary resource.

A cloud offloads tasks to a fog when it realizes that the latency requirements of the application cannot be achieved by the cloud computing environment. In this case, the cloud requires a local fog to provide service on its behalf. In this case, data is cached at the fog that is located close to the customer.

II. LITERATURE REVIEW

Zhang et al. [1] The proposed FEMTO algorithm effectively determines the FN feasibility and minimum energy consumption for task offloading services, resulting in a high and robust fairness level for energy consumptions.

Dang et al. [2] introduces FRATO (Fog Resource aware Adaptive Task Offloading) - a framework for the IoT-fog-cloud systems to offer the minimal service provisioning delay through an adaptive task offloading mechanism. FRATO-based service provisioning approaches reduce average delay significantly in systems with high rate of service requests and heterogeneous fog environment compared to existing solutions.

Malik et al. [3] propose a two phased task offloading algorithm to minimize task outages and energy consumption of IoT and fog nodes. In the first phase, we compute the minimum number of resources required for a task from the fog nodes and introduce variable sized VRUs. Simulation results show it outperforms existing techniques..

Tran-dang et al. [4] Reinforcement learning provides intelligent decision making for agents to respond effectively to environment, allowing RL to be applied to fog computing to improve performance.

Adhikari et al. [5] proposed DPTO strategy assigns a priority to each task based on its deadline and selects an optimal computing device based on its resource availability and transmission time from the IoT device.

Hussein et al. [6] proposed two nature-inspired meta-heuristic schedulers, namely ant colony optimization (ACO) and particle swarm optimization (PSO), are used to propose two different scheduling algorithms to effectively load balance IoT tasks over the fog nodes under communication cost and response time considerations. The experimental results of the proposed algorithms are compared with those of the round robin (RR) algorithm.

Swain et al. [7] propose a matching theory-based efficient task offloading strategy called METO that aims to reduce the total system energy and number of outages (number of tasks exceeding the deadline) in an IoT-fog interconnection network. As resource allocation involves

multiple criteria, their weights are derived using criteria importance though inter criteria correlation (CRITIC). Furthermore, to rank the alternatives we use the technique for order of preference by similarity to ideal solution (TOPSIS).

Raza et al. [8] propose an integer linear programming (ILP) model and a dynamic programming algorithm to maximize the number of successfully served IoT data tasks with satisfactory security requirements while minimizing the end-to-end transmission delay.

Chen et al. [9] proposed gradient algorithm with joint optimization of offloading ratio and transmission time improves convergence speed of traditional methods. Dynamic voltage scaling is integrated to achieve energy-optimal fog computation offloading. Results show superior energy consumption and completion time.

Yang et al. [10] a comprehensive analytical model is developed to evaluate energy efficiency in homogeneous fog networks and investigate the trade-off relationship between performance gains and energy costs. A MEETS algorithm is proposed to derive the optimal scheduling decision for a task node and multiple neighbouring helper nodes.

Mukherjee et al. [11] propose a latency-driven task data offloading problem to jointly optimize the number of tasks offloaded to the neighbouring fog nodes and communication resource allocation for the offloaded tasks to the remote cloud. Simulation results demonstrate that the proposed strategy is effective and scalable.

Aljanbhi et al. [12] propose a novel solution to the problem, where the IoT node has the choice to offload tasks to the best fog node or to the cloud based on the requirements of the applications and the conditions of the nearby fog nodes. IoT nodes can choose to offload tasks to fog nodes or the cloud based on requirements and conditions, with a Q-learning-based algorithm to select the optimal policy.

Deb et al. [13] propose a two-step distributed horizontal architecture for computation offloading in a fog-enabled Internet of Things environment—HD-Fog—to minimize the overall energy consumption, and latency while executing hard real-time applications. HD stands for the horizontal distribution of the tasks in the fog layer. The proposed HD-Fog scheme also indicates a reduction in energy consumption by 30% compared to traditional fog computing schemes.

Usman et al. [14] discuss the middleware technologies such as cloudlets, mobile edge computing, micro datacentres, self-healing infrastructures and delay tolerant networks for security provision, optimized energy consumption and to reduce the latency. The work introduces concepts of system virtualization and parallelism in IoT-Fog based systems and highlight the security features of the system.

Yamamoto et al. [15] propose an efficient task offloading method to improve the response time and to balance the processing load. The evaluation results show that the proposed method could effectively achieve the objective.

Misra et al. [16] The proposed scheme proposes a greedy-heuristic-based approach to efficiently solve the multi-hop task offloading problem in a fog-computing-based IoT architecture, reducing delay and energy consumption by 12% and 21%.

Khajafiy et al. [17] tackle the issue of fog congestion through a request offloading algorithm. The result shows that the performance of fogs nodes can be increased be sharing fog's overload over several fog nodes. The proposed offloading algorithm could have the potential to achieve a sustainable network paradigm and highlights the significant benefits of fog offloading for the future networking paradigm.

Liu et al. [18] The MTMHCO game is a non-cooperative game to model the competition among tasks for helpers, and an efficient distributed algorithm is developed to achieve an NE.

Kyung et al. [19] developed an analytic model for the opportunistic offloading probability that the task can be offloaded to the OFN, which can also be interpreted as the load distribution effect. Extensive simulation results are given to validate the analytic model and to demonstrate the performance of the opportunistic offloading probability.

Chebaane et al. [20] A flexible layer-oriented framework to offload time-critical application tasks from the application initiator.

Fang et al. [21] DRL-based resource allocation scheme to improve content distribution in FRAN using minimal delay model.

Wang et al. [22] Multiuser computational offloading scheme based on social trust, NOMA, and beamforming improves energy consumption, sum rate, and transmission latency.

III. METHODOLOGY

A. Task Offloading Optimization Using Evolutionary Algorithm

For real-time sensitive IoT applications, solving the IoT fog task offloading challenge is critical. Network latency, capacity, nodes order processing, and present fog node demand must all be addressed when distributing data generated by IoT devices and fog nodes. Based on the network variables and current fog node load, the task offloading solution must identify a fog device target that is guaranteed to fulfil the required QoS limits, particularly in terms of response time. As the number of sensors and fog nodes grows, the complexity of this NP-hard process increases exponentially.

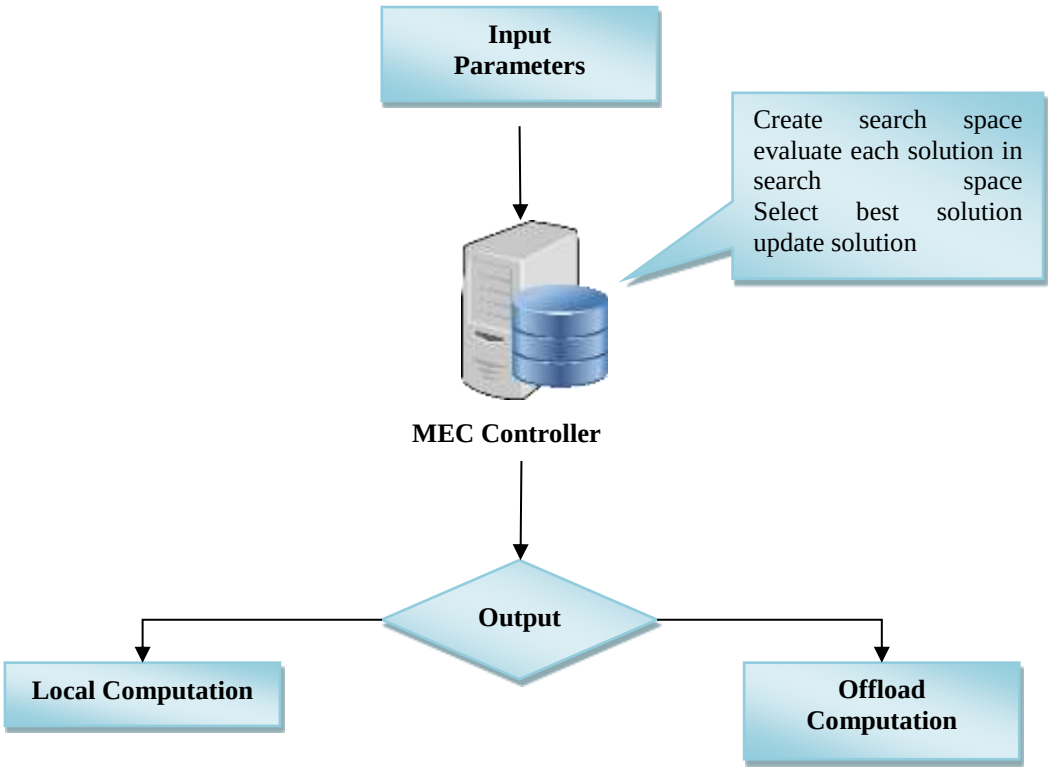

Fig. 2: Offloading using Evolutionary Optimization Algorithm

For tackling optimization issues, evolutionary algorithms are well-known [12]. They generate a search space with many solutions, each of which is made up of all possible properties that meet requirements. The fitness function is used to assess the value of each solution.

B. Optimal Offloading Decision

In this research work, we have presented a deep learning model for optimal offloading of tasks from IoT devices to edge/fog nodes. The model is designed in two basic steps: task prediction and optimal offloading. The algorithm of designed model is presented as below in Algorithm 1.

Algorithm 1: Task Prediction Model and Optimal Offloading Decision

1) Begin
2) Initialize No. of users (U), No. of tasks (T), Queue (Q)
3) For i=1: U
4) For j=1: T
5) Train Neural Network and predict arrival of next task
6) Maintain Q
7) Offload using MOGWO
8) End for
9) End for
10) End

C. Task Prediction Model

As a result, we could forecast the tasks which will be created within next networking timestamp of devices depending on their history, as well as load the service packages ahead of time before the actual tasks arrive (e.g., allocating the most appropriate computational nodes prior with the decision model). For example, insert D_nt as the input data into the trained neural network estimation method to forecast a next time slot task information t for every node n with task data D_nt. As a result, the predictor model's optimizing target is minimize (S{d,o,b}. In an actual case, as illustrated in Figure 3.3, we may use the prediction system to anticipate the data for a future task, make a choice, and allocation computing resources again for task.

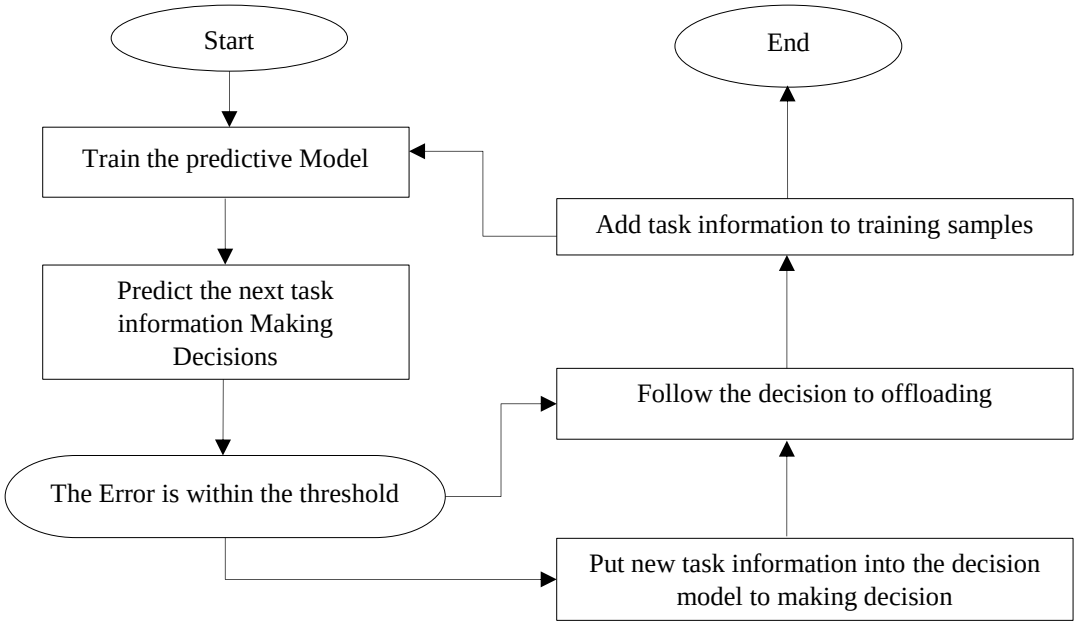

Fig. 3: Flow chart of task prediction

Algorithm 2: Task Prediction (neural network)

1) Begin
2) Setup input parameters
3) Initialize neural network units
4) For i=1: No. of epochs
5) Training of Network
6) Evaluated Loss Function (L_f)
7) If L_f is min
8) Return trainable parameters
9) Else
10) Adjust internal parameters
11) End if
12) End for
13) End

Figure 4 shows that the candidate solution eventually falls inside the randomized circle described by α, β and δ. The remaining contenders then modify their locations near the prey at random, guided by the best current 3 wolves. They begin by searching for prey location information in a disorganized manner before focusing on assaulting the target.

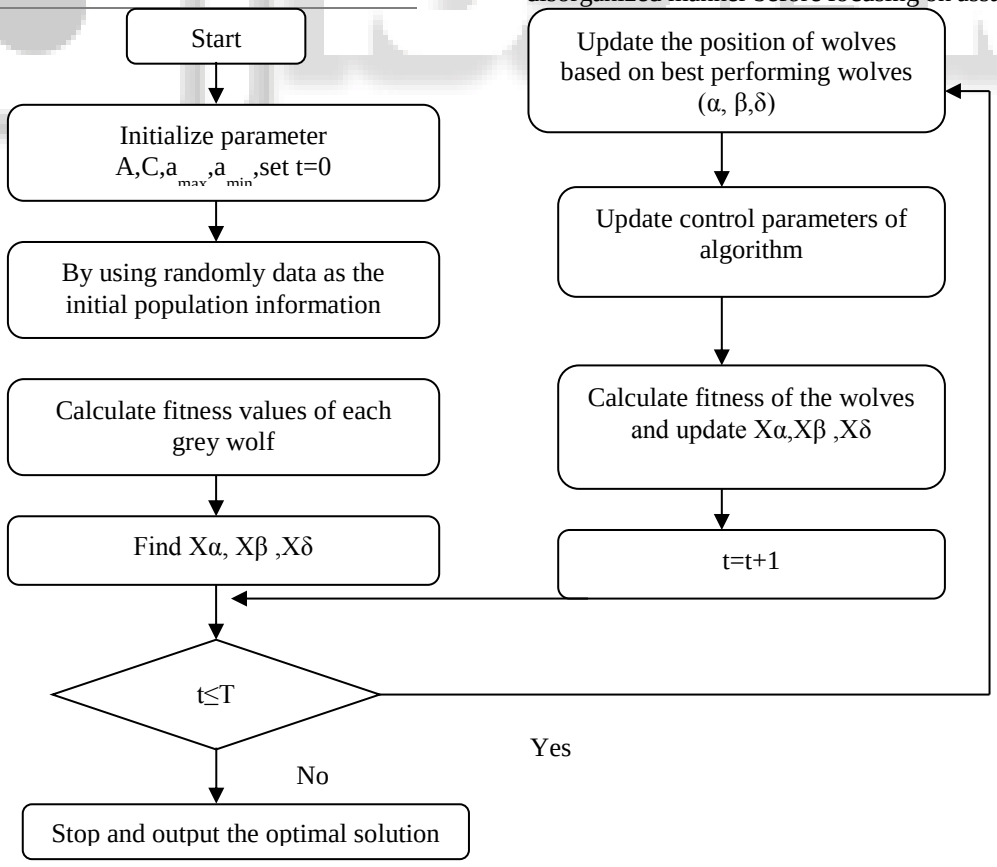

Fig. 4: Flowchart of Grey Wolf Optimization

Algorithm 3: Optimal Offloading (GWO Algorithm)

1) Begin, Randomly initialize the population of grey wolves Xi (i=1, 2,…,n)
2) Initialize variables
3) Calculate the fitness of each member of the population
4) Xα=member with the best fitness value
5) Xβ=second best member (in terms of fitness value)
6) Xδ=third best member (in terms of fitness value)
7) For t = 1 to Max_number_of_iterations:
8) Update the position of all the omega wolves
9) Update variables
10) Calculate Fitness of all search agents
11) Update Xα, Xβ, Xδ.
12) End For
13) return Xα
14) End

IV. RESULTS AND DISCUSSIONS

To evaluate the effectiveness of the proposed algorithms, the proposed algorithm is implemented using MATLAB R-2020a. The simulations were conducted on an Intel i5, 3.7 GHz PC with 8 GB RAM

A. Simulation Details

In this work, fog-cloud computing system consisting of mobile users and multiple BSs. Each BS equipped with fog servers. The parameters of the simulation are shown in Table 4.1. Heuristic request offloading based task offloading algorithm can find a local optimum in polynomial time, and resource scheduling can be solved.

Parameters	Value
Number of mobile users	2000
Bandwidth of fog computing	12MB/s
Bandwidth of local	300MB/s
Data Size	250kb-1Mb
Ideal delay of request	0.4-0.6s

Table 1: System Parameters Used

B. Performance Analysis

The tests are carried out to assess the suggested MOGWO task offloading method in terms of many evaluation measures, including energy consumption and average delay/response time (RT).

C. Result Analysis

In this section, we have presented a result analysis for two cases. One for variable number of IoT nodes/users and another for variable size of tasks. The result was evaluated in terms of response time as well as energy used.

Table 2 shows the energy usage of proposed algorithm with variable task size. In this case number of IoT nodes/users are fixed and task size is varied. The task size was taken from 250Kb to 1000Kb. The result analysis presented in Figure 5 shows that with increases size of task energy utilization also increases.

Task Size (in Kb)	Energy (in J)
250	67.89
500	135.77
750	203.66
1000	271.55

Table 2: Energy Usage with Variable Task Size

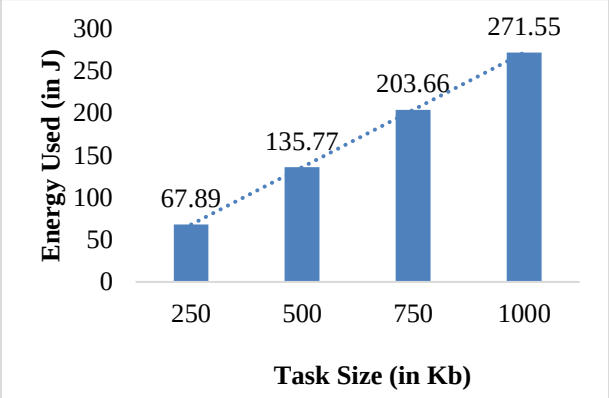

Fig. 5: Energy Usage with Variable Task Size

No. of Nodes	Response Time
250	5.78
500	5.86
750	6.3
1000	6.3

Table 3: Response Time with Variable Nodes

Table 3 shows the response time of proposed algorithm with variable number of users. In this case number of users are varied and task size is fixed. The number of users was taken from 250-2000. The result analysis presented in Figure 6 shows that with increases size of users, response time also increases.

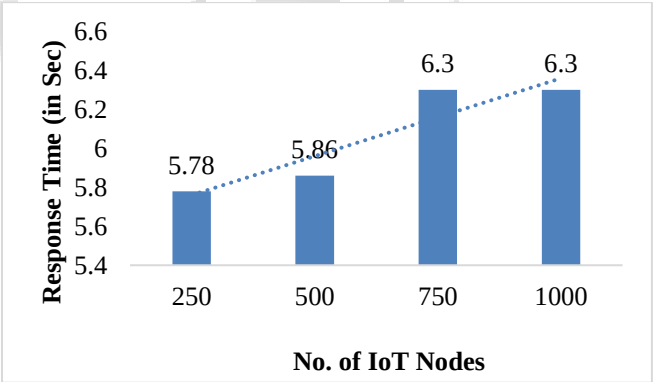

Fig. 6: Response Time with Variable User

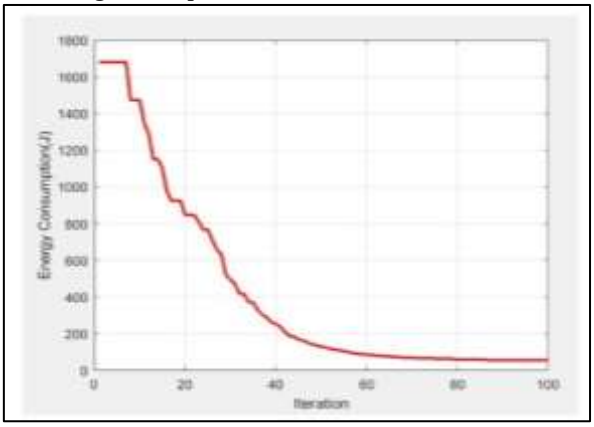

Fig. 7: Energy Consumption Vs Iterations

Figure 7 depicts the energy consumption as a function of iterations. The graph clearly shows that the method converges in a short number of iterations. Every iteration's energy consumption, measured in joules, is noted. The suggested MOGWO method is faster than the PSO and ACO algorithms in terms of convergence: the optimal value is attained after iteration number 60. In comparison to the ACO and PSO algorithms, this finding indicated that the suggested MOGWO algorithm investigated the search space and found optimal value with a fair level of energy consumption.

V. COMPARATIVE ANALYSIS

Figure 8 compares the average response times of MOGWO with ACO and PSO offloading techniques, as the total number of IoT nodes increases. The fig shows that as the number of nodes increases, the average response time also increases for all the offloading techniques. However, MOGWO has significantly lower average response times compared to ACO and PSO. For example, when there are 250 IoT nodes, MOGWO's average response time is 5.78 seconds, while ACO and PSO have response times of 80 and 78 seconds, respectively. As the number of nodes increases to 1000, MOGWO's response time remains the same at 6.3 seconds, while ACO and PSO's response times increase to 85 and 88 seconds, respectively.

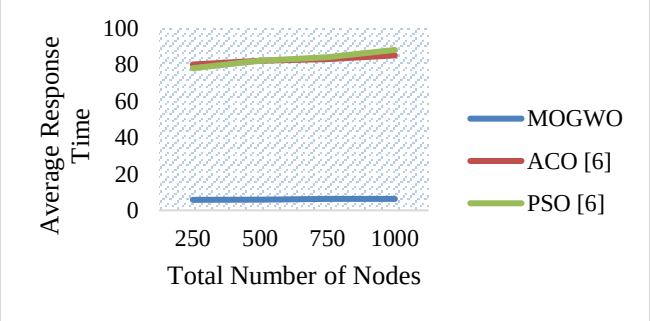

Fig. 8: Average Response Time Comparison with Total Number of Nodes

VI. CONCLUSION

This study proposed and evaluated a task prediction and optimal offloading decision method for fog computing cloud networks. Therefore, this work presented a multi-objective GWO optimization algorithm for task offloading in fog computing. The proposed method considered two evaluation parameters, including average response time and energy consumption. The proposed method was simulated, and the results showed improvements in average response time, energy consumption, and load balancing measures, demonstrating the effectiveness of the method. From the presented results, it can be concluded that the MOGWO offloading techniques have significantly lower average response times compared to ACO and PSO, especially as the number of IoT nodes increases.

Integration of multiple optimization techniques: Future work could explore the integration of different optimization techniques, such as combining evolutionary algorithms and deep reinforcement learning, to improve the efficiency and effectiveness of task offloading in cloud computing systems.

REFERENCES

[1] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "FEMTO: Fair and Energy-Minimized Task Offloading for Fog-Enabled IoT Networks," IEEE Internet Things J., vol. 6, no. 3, pp. 4388–4400, 2019, doi: 10.1109/JIOT.2018.2887229.

[2] Tran-dang, D. Kim, and S. Member, "FRATO : Fog Resource Based Adaptive Task Offloading for Delay-Minimizing IoT Service Provisioning," vol. 32, no. 10, pp. 2491–2508, 2021.

[3] U. M. Malik, M. A. Javed, J. Frnda, and J. Nedoma, "SMRETO: Stable Matching for Reliable and Efficient Task Offloading in Fog-Enabled IoT Networks," IEEE Access, vol. 10, pp. 111579–111590, 2022, doi: 10.1109/ACCESS.2022.3215555.

[4] H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim, "Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues," J. Commun. Networks, vol. 24, no. 1, pp. 83–98, 2022, doi: 10.23919/JCN.2021.000041.

[5] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A Deadline and Priority-Aware Task Offloading in Fog Computing Framework Leveraging Multilevel Feedback Queueing," IEEE Internet Things J., vol. 7, no. 7, pp. 5773–5782, 2020, doi: 10.1109/JIOT.2019.2946426.

[6] M. K. Hussein and M. H. Mousa, "Efficient Task Offloading for IoT-Based Applications in Fog Computing Using Ant Colony Optimization," IEEE Access, vol. 8, pp. 37191–37201, 2020, doi: 10.1109/ACCESS.2020.2975741.

[7] C. Swain et al., "METO: Matching-Theory-Based Efficient Task Offloading in IoT-Fog Interconnection Networks," IEEE Internet Things J., vol. 8, no. 16, pp. 12705–12715, 2021, doi: 10.1109/JIOT.2020.3025631.

[8] M. M. Razaq, B. Tak, L. Peng, and M. Guizani, "Privacy-Aware Collaborative Task Offloading in Fog Computing," IEEE Trans. Comput. Soc. Syst., vol. 9, no. 1, pp. 88–96, 2022, doi: 10.1109/TCSS.2020.3047382.

[9] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-Optimal Dynamic Computation Offloading for Industrial IoT in Fog Computing," IEEE Trans. Green Commun. Netw., vol. 4, no. 2, pp. 566–576, 2020, doi: 10.1109/TGCN.2019.2960767.

[10] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M.-T. Zhou, "MEETS: Maximal Energy Efficient Task Scheduling in Homogeneous Fog Networks," IEEE Internet Things J., vol. 5, no. 5, pp. 4076–4087, 2018, doi: 10.1109/JIOT.2018.2846644.

[11] M. Mukherjee et al., "Latency-Driven Parallel Task Data Offloading in Fog Computing Networks for Industrial Applications," IEEE Trans. Ind. Informatics, vol. 16, no. 9, pp. 6050–6058, 2020, doi: 10.1109/TII.2019.2957129.

[12] S. Aljanabi and A. Chalechale, "Improving IoT Services Using a Hybrid Fog-Cloud Offloading," IEEE Access, vol. 9, pp. 13775–13788, 2021, doi: 10.1109/ACCESS.2021.3052458.

[13] P. K. Deb, S. Misra, and A. Mukherjee, "Latency-Aware Horizontal Computation Offloading for Parallel Processing in Fog-Enabled IoT," IEEE Syst. J., vol. 16, no. 2, pp. 2537–2544, 2022, doi: 10.1109/JSYST.2021.3085566.
[14] S. U. Jamil, M. A. Khan, and M. Ali, "Security Embedded Offloading Requirements for IoT-Fog Paradigm," in 2019 IEEE Microwave Theory and Techniques in Wireless Communications (MTTW), 2019, vol. 1, pp. 47–51. doi: 10.1109/MTTW.2019.8897219.
[15] T. Yamamoto, R. Yamamoto, S. Ohzahata, and T. Kato, "Delay-based Task Offloading for Fog Computing in IoT Networks," in 2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), 2021, pp. 1–2. doi: 10.1109/ICCE-TW52618.2021.9602895.
[16] S. Misra and N. Saha, "Detour: Dynamic Task Offloading in Software-Defined Fog for IoT Applications," IEEE J. Sel. Areas Commun., vol. 37, no. 5, pp. 1159–1166, 2019, doi: 10.1109/JSAC.2019.2906793.
[17] M. Al-Khafajiy, T. Baker, A. Waraich, O. Alfandi, and A. Hussien, "Enabling High Performance Fog Computing through Fog-2-Fog Coordination Model," in 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), 2019, pp. 1–6. doi: 10.1109/AICCSA47632.2019.9035353.
[18] Z. Liu, X. Yang, K. Wang, Y. Yang, and Z. Shao, "Computation Offloading Game for Multi-Task Multi-Helper Fog Networks," in 2019 IEEE Global Communications Conference (GLOBECOM), 2019, pp. 1–6. doi: 10.1109/GLOBECOM38437.2019.9013933.
[19] Y. Kyung, "Performance Analysis of Task Offloading With Opportunistic Fog Nodes," IEEE Access, vol. 10, pp. 4506–4512, 2022, doi: 10.1109/ACCESS.2022.3141199.
[20] A. Chebaane, S. Spornraft, and A. Khelil, "Container-based Task Offloading for Time-Critical Fog Computing," in 2020 IEEE 3rd 5G World Forum (5GWF), 2020, pp. 205–211. doi: 10.1109/5GWF49715.2020.9221486.
[21] C. Fang et al., "Deep-Reinforcement-Learning-Based Resource Allocation for Content Distribution in Fog Radio Access Networks," IEEE Internet Things J., vol. 9, no. 18, pp. 16874–16883, 2022, doi: 10.1109/JIOT.2022.3146239.
[22] [22] L. Wang, M. Guan, Y. Ai, Y. Chen, B. Jiao, and L. Hanzo, "Beamforming-Aided NOMA Expedites Collaborative Multiuser Computational Offloading," IEEE Trans. Veh. Technol., vol. 67, no. 10, pp. 10027–10032, 2018, doi: 10.1109/TVT.2018.2853675.

