Grey Wolf Optimizer-based Task Scheduling for IoT-based Applications in the Edge Computing

Aram Satouf
Department of Computer Engineering
Bahcesehir University
Istanbul, Turkiye
email: aram.satouf@bahcesehir.edu.tr

Ali Hamidoglu
Department of Mathematics
Bahcesehir University
Istanbul, Turkiye
email: ali.hamidoglu@eng.bau.edu.tr

Omer Melih Gul
Department of Computer Engineering
Bahcesehir University
Istanbul, Turkiye
email: omermelih.gul@bau.edu.tr

Alar Kuusik
School of Information Technologies
Tallinn University of Technology
Tallinn, Estonia
email: alar.kuusik@taltech.ee

2023 Eighth International Conference on Fog and Mobile Edge Computing (FMEC) | 979-8-3503-1697-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/FMEC59375.2023.10306148

***Abstract*—The growing data volume generated by IoT devices places considerable resource constraints on traditional cloud data centers, compromising their ability to cater to delay-sensitive IoT applications. The emergence of cloud-fog computing offers a potential solution, by extending cloud resources to the network edge. Yet, task scheduling in the cloud-fog environment introduces new challenges. Our study presents a semi-dynamic real-time task scheduling algorithm developed specifically for the cloud-fog environment, which efficiently allocates tasks while minimizing energy consumption, cost, and makespan. We utilized a modified grey wolf optimizer to optimize task allocation based on parameters like task length, resource requirements, and execution time. Compared to existing methods, including genetic algorithm, our algorithm demonstrates superior performance in terms of makespan, total execution time, cost, and energy consumption.**

***Index Terms*—Internet of Things (IoT); task scheduling; fog and edge computing; optimization; energy consumption**

I. Introduction

Industrial edge computing is helping several industries manage massive amounts of data. Edge gateways provide many advantages for any industry that collects IoT data. Some of these advantages include improved bandwidth, reduced latency, cost-efficiency, and reliability.

With the dramatic increase in data volume due to the mounting number of interconnected devices. Cloud data centers, being the traditional backbone for data processing in the IoT infrastructure, are now finding it increasingly challenging to grapple with the rising tide of data and the real-time processing demands of many IoT applications. These hurdles arise from the latency intrinsic in transferring data to and from the cloud for processing. This dilemma has driven the exploration of alternative models capable of facilitating efficient and timely data processing at a location closer to the data sources. While the proximity of data centers to computations has diminished the latency concern to a significant extent, the need for edge computing remains pertinent in specific use cases. For instance, in situations where milliseconds matter, such as applications relying on real-time analytics or mission-critical decision-making, the delay introduced by transmitting data back and forth to distant cloud data centers can still impact performance [1]. Emerging as a promising alternative is the cloud-fog computing model. This model challenges the conventional cloud-based paradigm, which processes data in centralized data centers, by extending computational capabilities to the network's edge, closer to the data generation points [2]. This model's strategic distribution of computational tasks across the network significantly mitigates latency and enables more adept data processing. However, transitioning to a cloud-fog environment heralds its own set of challenges. The decentralization of computational resources calls for an efficient system to manage and allocate these resources, ensuring optimal system performance. Herein lies the importance of task localization and scheduling. Task scheduling, in this context, entails identifying the best location (either cloud or fog nodes) and the most suitable timing for task execution, considering various factors such as computational resource availability, network latency, and the unique requirements of each task [3]. Crucially, efficient task scheduling is also pivotal for optimizing energy consumption. As the number of interconnected IoT devices increases, the energy required for data processing escalates, posing significant environmental and economic challenges. By ensuring tasks are executed energy-efficiently, we can significantly reduce overall energy consumption, contributing to more sustainable and cost-effective operations. Additionally, task scheduling plays a key role in minimizing operational costs and reducing the makespan – the total time required to process all tasks, which is of particular importance for latency-sensitive IoT applications.

A. Motivation

The complexity of task scheduling is the primary motivation for our study. We sought to design a task scheduling algorithm tailored for cloud-fog environments, focusing on efficient task

Identify applicable funding agency here. If none, delete this.

allocation, energy consumption, cost, and makespan minimization. Our algorithm leverages a modified version of the grey wolf optimizer (GWO) [4], and performs better across key metrics compared to existing methods. This research offers a significant contribution to task scheduling at a cloud-fog environment by highlighting potential of our approach to handle the growing volume of IoT data efficiently.

B. Main Contributions

The primary innovation of this paper lies in the development of a novel semi-dynamic real-time task scheduling algorithm, purposefully tailored for cloud-fog computing environments. Distinctive from conventional methods, our proposed algorithm harnesses a customized adaptation of the GWO to facilitate efficient task allocation.

The algorithm is built upon a careful blend of essential elements like the number of tasks, resource needs, and execution time. This combination ensures a balance between optimizing resource use, reducing energy, costs, and makespan.

C. Organization of the paper

The rest of the paper is organized as follows. Section II presents the related work. Section III first provides the system model and then defines the problem precisely. Section IV proposes a grey wolf optimizer algorithm for this problem. Section V evaluates the numerical performance of the grey wolf optimizer algorithm with the genetic algorithm. Section VI concludes the paper. Section VII provides the limitation of the work and suggest enhancement to the work.

II. Related Work

The IoT-Edge-Fog Computing model provides tri-level framework for decentralized, time-sensitive computing. The increasing need for real-time data processing and decision modeling has made it challenging to distribute tasks across remote Edge Computing nodes. Existing task allocation algorithms like Round Robin, Minimum Completion Time, and Min-Max, face issues related to energy efficiency, latency, and error rates. In response to these challenges, an algorithm based on meta-heuristic principles inspired by quantum computing was introduced in [5]. This approach aims to establish an effective task allocation for real-time IoT application, predicting ideal locations of computing nodes for real-time devices.

In [6], a Hybrid Flamingo Search with a Genetic Algorithm (HFSGA) is utilized as a meta-heuristic strategy to enhance task scheduling and achieve cost minimization. This approach is then benchmarked against other renowned algorithms using seven essential optimization test functions, with the significance assessed via Friedman Rank Test. This model results in improvements in the Percentage of deadline satisfied tasks, makespan, and cost. Notably, the proposed algorithm outperforms other methods such as ACO, PSO, GA, Min-CCV, Min-V and Round Robin (RR) approach in task scheduling efficiency. The approach showed improved results in the Percentage of Deadline Satisfied Task (PDST), makespan, and cost. However, it continues to exhibit issues of limited precision and a sluggish rate of convergence.

In [8] a meta-heuristics-based latency-aware scheduling solution was proposed for smart home energy management that combines cloud and fog computing. The Tabu search is a well-known heuristic method because of its memory, speed, and extension of optimization. As a result, an improved Tabu search technique utilizing fruit fly optimization and approximate closest neighbor is suggested. The proposed method is tested by modeling a case study and running the algorithm with the performance targets of execution time, latency, allocated memory, and cost function. The proposed method uses tabu search, GA, PSO, and simulated annealing.

For delay-sensitive and energy-limited applications such as virtual reality and mobile 3-D gaming, there is a considerable need to decrease energy usage and service delay in homogeneous fog networks. The research [9] introduces a multi-layer analytical structure intended to strike balance between service latency and energy consumption. By employing various optimization algorithms, this study developed a balanced task scheduling method aimed at minimizing energy use, service delay, and disruption. However, heterogeneous capabilities of nodes were not considered in the study.

In [10], a novel method aimed at optimizing task scheduling for Bag-of-Tasks applications within Cloud-Fog environments was introduced, focusing specifically on execution time and operational costs. The method, known as TCaS, was put to the test using 11 distinct datasets. In terms of balancing completion time and operational cost, the Bee Life Algorithm (BLA) and Modified Particle Swarm Optimization (MPSO) outperformed each other by 15.11% and 11.04%, respectively. However, a limitation was the higher communication costs incurred due to offloading all tasks to the distant cloud.

III. System Model and Problem Definition

This section first introduces the task scheduling problem in fog/edge computing. Then, it defines the problem precisely.

The system model demonstrates a three-layer architecture for a fog and cloud computing environment. The cloud layer is the topmost layer. The idea of a cloud layer for computing and storage. Fog computing, which consists of routers, switches, and gateways, is the middle layer. Instead of moving everything to the cloud, these devices deliver IoT services at the network edge. The architecture's last layer is made up of end-user devices that generate a lot of data. Multiple virtual machines (VMs) on the cloud are vital for high-demand IoT applications, such as real-time data analytics or machine learning, which require extensive computational resources. Furthermore, the use of multiple VMs enables effective resource management, enhancing security, scalability, and fault isolation. This is particularly beneficial in complex IoT systems like smart cities or large-scale industrial deployments, where numerous IoT devices and services operate concurrently [11]. The whole scheduling procedure in the IoT context is shown in Fig. 1 [12]. Applications used by IoT end users send requests to a central node, where a request

evaluator prioritizes various tasks. Task scheduling considers various factors, including resource demand, communication requirement such as throughput and latency, and computation size. The task scheduler arranges the tasks to allow them to be completed within the limitations imposed by the problem.

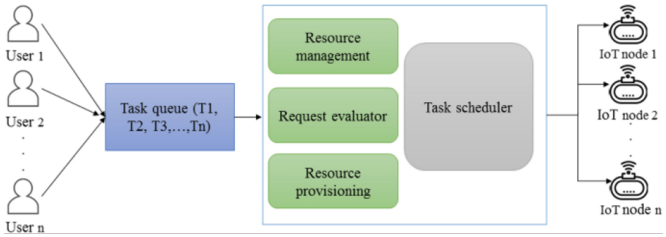

Fig. 1. An overview of the IoT job scheduling process

S_{cn} and S_{fn} denote the set of cloud nodes and the set of fog nodes, respectively. The total number of nodes is thus represented by $S = S_{cn} \bigcup S_{fn}$. Assume there are m virtual machines (VMs), $VM = \{vm_1, vm_2, vm_3, ..., vm_i\}$ that are used to execute the n Tasks $T = \{T_1, T_3, T_3, \ldots, T_n\}$. where each parameter T_i is described by a set of attributes

$$T_j = \{TS_j, TL_j, Type_j, TD_j\} \tag{1}$$

where

- TS_j is task size (in bits);
- TL_j is task length (in MI (Millions of Instructions));
- $Type_j$ is the type of task whether it is: normal, moderate, or critical;
- TD_j is the task deadline that must be adhered to in order to complete the task.
- X_{ij} is the assignment of task T to the fog node.

Fog nodes handle the incoming tasks first, followed by cloud nodes and the outcomes of prior operations. The challenge is to distribute workloads across processing units in a way that enhances system performance while utilizing the least amount of energy and computational cost. These tasks are of varying lengths, and the VMs' bandwidth, RAM, and CPU usage are variable. The cloud broker learns about the services required to complete the tasks it has been given, and after finding the needed resources, schedules the tasks on those resources. To distribute the tasks to a suitable fog node based on the resources of the fog node, we analyze execution time, cost, makespan, and energy consumption involved in distributing the requested tasks to the appropriate fog node. Once we have defined that, we select the most suitable node and assign the task to it. Following notions are useful in the rest of the paper:

- *Computation Cost:* Every task that one computing node completes carries a monetary cost. This computation cost is different for fog and cloud nodes, due to the differing infrastructural and operational characteristics of these two types of nodes. In this study, we have chosen to use a unified cost model for both cloud nodes and fog nodes. This decision is based on the need to maintain the simplicity and tractability of our model. Processing and memory costs, which together make up the computing cost of a given task, can be calculated as the sum of CPU and RAM allocated to the machine:

$$(Cost_{vm})_{total} = (Cost_{vm})_{CPU} + (Cost_{vm})_{RAM} \tag{2}$$

- *Execution Time:* The execution time of processing task (T) on the fog node or the cloud (S_{cn} and S_{fn}.) is:

$$EX_{vm} = \frac{TL}{CC} \tag{3}$$

where TL is task length and CC is computation capacity.

- *Makespan:* The makespan is the maximum execution time (Ex) of a resource. The following is how makespan is:

$$Makespan = \max\{EX^{max}_{vm1}, EX^{max}_{vm2}), ..., EX^{max}_{vmj}\} \tag{4}$$

- *Energy Consumption:* Energy consumption by a fog node can be accurately described as a linear relationship of CPU utilization [13] [14]. While our focus lies in optimizing computational energy in fog nodes, we acknowledge that the model overlooks communication power costs. This aspect, although substantial in scenarios with prevalent wireless communication, falls beyond this study's scope. Future research could strive to develop a more comprehensive model, incorporating both computational and communication power costs. By considering idle energy consumption and CPU utilization (u), the energy consumption of a computing node (P_i) is defined as

$$P_i(u) = w \times P_{max} + (1 - k) \times P_{max} \times u \tag{5}$$

The maximum amount of energy used by a host while it is operating at full capacity (100 percent utilization) is P_{max}, and the amount of power used by an idle host is w. You can figure out how much energy a fog landscape with n nodes uses overall by using: $\sum_{i=1}^{n} Pi(u)$ The weighted sum of the QoS parameters, total energy consumption, and makespan is used to determine the QoS score assigned to X_ij. To make the dynamic modified Weighted Sum Method (MWSM) that is appropriate for our task, we changed the Weighted Sum Method (WSM). [15]

$$N_{ij}^{MWSM-Score} = w_E \times E_{total}(X_{ij}) + w_L \times Makespan(X_{ij}) + wf \times (Cost_{vm})_{total}(X_{ij}),\ \forall j \in \{1, \ldots, m\}$$

where $N_{ij}^{MWSM-Score}$ denotes the QoS score of X_{ij}.

The fog node is more suitable for carrying out task T for lower $N_{ij}^{MWSM-Score}$ values. The weighted average of the QoS parameters, total energy consumption, and makespan is used to determine the QoS score assigned to X_{ij}. The weight factors' total must be equal to 1:

$$w_E + w_L + w_F = 1 \tag{6}$$

IV. Proposed Approach

This section presents meta-heuristics and optimization algorithms for the task scheduling problem at hand.

The Grey Wolf Optimizer (GWO) is a swarm intelligence-based artificial intelligence (AI) algorithm that was introduced

by [4]. It draws inspiration from the behavior of grey wolves in nature as they collaborate to hunt for prey efficiently. By mimicking the hierarchical structure and roles within a wolf pack, the GWO algorithm organizes its members into four distinct groups. These groups, namely Alpha α, Beta β, Delta δ wolves, each play a vital role in the hunting process. Among them, the Alpha wolves represent the best solutions found so far during the optimization process. They guide the movement of the pack towards more promising regions of the search space. The Beta wolves assist the Alpha wolves and are the second-best solutions in the population. They provide perspectives to the Alpha wolves to improve leadership. The Delta wolves are the remaining lowest-ranking members of the pack that follow the Alpha and Beta wolves during hunting.

This hierarchical division enables collaboration between candidate solutions to balance exploration and exploitation. The Alpha wolves exploit the best area discovered, while the Beta and Delta wolves explore the search space for better solutions. The constant communication and feedback between these groups enable the movement of the pack towards the optimal region over iterations [4]. Through this approach, the GWO algorithm leverages the collective intelligence of a wolf pack to optimize and solve complex problems.

This section introduces a permutation-based, semi-dynamic real-time task scheduling algorithm that is built based on [16].

Our proposed solution is given in Algorithm 1.

Algorithm 1 Scheduling Rounds Algorithm

Input: Number of scheduling rounds (n_r), set of real-time tasks T, set of VM, population size (p) and maximum number of iterations (t_{max})
Initialization: Initialize cumulative schedule S_c as empty.
Algorithm:
for $k = 1 : n_r$
  if $(k == 1)$ **then**
    Create set of real-time tasks T that have $BAT = 0$
    Schedule the current list of real-time tasks using Grey Wolf Optimization algorithm
  else
    Create a list of real-time tasks that have $((k-2)\times t) < BAT \leq ((k-1)\times t)$
    Schedule the current list of real-time tasks using Grey Wolf Optimization algorithm
  end if
  Update the current resources available to VMs
  Merge the current round's best schedule S_k with the cumulative schedule S_c.
end for
Output: Return the cumulative task schedule S_c.

This proposed algorithm starts by initializing an empty schedule (S_c). Then it goes through each scheduling round, for each round it checks if this is the first round. If it is, it gathers all tasks that are available at the beginning (BAT = 0) and schedules them using GWO algorithm. If it's not the first round, it gathers all tasks that become available in this round and schedules them. After scheduling the tasks, it updates the available resources of the VMs and merges the new schedule with the cumulative schedule. The algorithm repeats these steps until it has gone through all scheduling rounds. Finally, it returns the cumulative schedule.

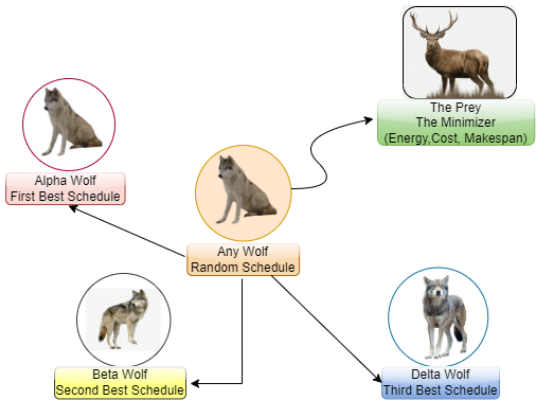

Fig. 2. Application of GWO in scheduling rounds algorithm to determine the optimal schedule in terms of cost, energy, and duration.

In the Scheduling Rounds algorithm, GWO algorithm is called to schedule each set of tasks that become available in each round. The factor a is a coefficient used in GWO algorithm. As illustrated in Fig. 2, GWO algorithm starts by choosing random candidates where the positions of the α, β and δ wolves are initialized according to the cost function. The algorithm then enters a loop that runs for a maximum of t_{max} iterations. In each iteration, it calculates the fitness of each wolf (solution) and updates the positions of α, β and δ wolves if a better solution is found. It also updates the position of each wolf to move closer to the positions of α, β and δ wolves. If a stopping condition is met (like reaching a certain number of iterations or finding a solution that's good enough), the algorithm stops early. Finally, it uses α wolf's position as the best task-VM mapping for this set of tasks (S_k) and returns this as the result.

V. Numerical Results

This section provides numerical comparison of the proposed approach with GA algorithm. The simulated experiments replicate real-world IoT scenarios like smart transportation systems and industrial IoT deployments. The chosen parameters of Fog/Cloud VMs, task attributes, and workload volumes reflect the characteristics of high-demand IoT applications, as the experimental setup is carefully designed to simulate such real-world conditions. For instance, the scenarios with varying VM compositions can model a smart city's hybrid infrastructure, while task attributes capture IoT workloads involving streaming data and low-latency needs.

To assess our proposed algorithm, we ran experiments using GA as a benchmark. GA is renowned for solving intricate combinatorial problems, making it a suitable representative technique among various existing methods. Its aptitude for diverse solution exploration and multi-objective optimization aligns with IoT task scheduling's intricacies.

In our experimental setup, we combined simulation-based experimentation and algorithmic evaluations. A Python-based simulation framework was developed to accurately simulate the cloud-fog task scheduling environment. This framework utilized randomly generated tasks and VMs with attributes relevant to cloud and fog infrastructures. Various parameters, including task lengths, CPU and memory requirements, VM types, and costs, were considered in the simulation. Custom-defined functions calculated execution times, execution costs, makespan, and energy consumption for each task-VM allocation. These instruments were carefully designed to capture the diverse characteristics of real-world cloud-fog environments, ensuring the accuracy and relevance of the collected data.

To facilitate meaningful comparisons, an array of simulation experiments was meticulously designed and implemented. These simulations were carried out within the environment provided by Amazon Elastic Compute Cloud (EC2), a renowned and versatile cloud-based virtual machine platform. This choice was underpinned by EC2's flexibility and computational prowess, indispensable for handling the intricate computations associated with metaheuristic algorithms. Notably, the EC2 instance of choice for the simulations was an on-demand t2.xlarge type, selected after thorough evaluation.

In the setup, we considered a different (N) number of tasks, each with attributes such as length of the task (L), required memory (REQ-MEM), required CPU (REQ-CPU), input file size (IFS), and output file size (OFS). We also considered the number of cores, CPU cost (CPU-COST), RAM cost (RAM-COST), bandwidth (BW), available memory (A-MEM), and available CPU (A-CPU).

Each VM was either a Fog type or a Cloud type. For our simulations and evaluation of impact of VM type on task scheduling, we adopted three scenarios in the conducted experiments: 10 Fog VMs and 5 Cloud VMs, 20 Fog VMs and 10 Cloud VMs, and 30 Fog VMs and 15 Cloud VMs. However, the use of VMs in our experiments serves to abstract the underlying hardware complexity and allows us to focus on the task scheduling aspect in heterogeneous environments. This experimental design is intended to provide theoretical insights. To extend our research, we generated different synthetic task datasets with various sizes ranging from $N = 50$ to $N = 400$ tasks, to study the impact of task volume on our scheduling model. Each task also had a unique broker arrival time (BAT).Different weights were assigned to Eq.(8) to reach balance of the QoS score. We have also initialized N and D variables that represent the number of agents (population size) and the dimension of the problem (number of variables), respectively.

A. Convergence Analysis

This subsection offers a detailed examination of the convergence properties of the scheduling algorithms employed, GWO and GA. The experimental setup includes 10 fog VMs and 5 cloud VMs which are tasked with executing a set of 100 tasks. The heart of the convergence analysis lies in the assessment of the fitness value. This fitness value is a representation of

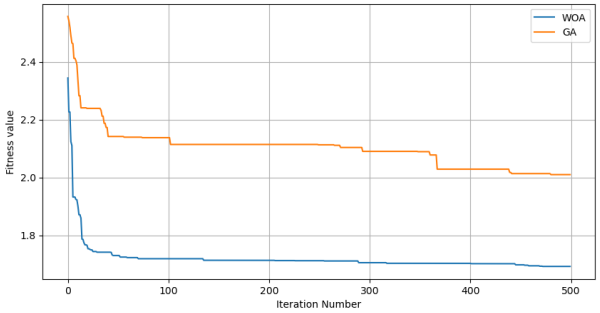

Fig. 3. Convergence Analysis of GWO and GA

the equilibrium attained among the makespan, the cumulative execution cost, and the total energy consumed. As shown in Fig. 3, the GWO and the GA exhibit distinct convergence characteristics. Both algorithms converge, but GWO reaches a better fitness value 1.86 compared to GA. This suggests that GWO is more effective at minimizing the makespan, cost, and energy for this specific task scheduling problem in fog-cloud computing environments. As the number of iterations increases, the spectrum of fitness values for the GWO begins to tighten. This pattern suggests that the algorithm's populations are steadily moving towards solutions that are either optimal or in close proximity to the optimal state.

B. Fitness Analysis

The employed fitness function has been crafted to attain an optimal balance among the makespan, total execution cost, and energy consumption of tasks. Recognizing our task scheduling problem as a minimization optimization task, a smaller fitness value indicates a more effective equilibrium between the makespan, execution cost, and energy consumption. In order to evaluate the performance of different scheduling algorithms, an experiment was carried out using 15 VMs - a combination of 10 fog VMs and 5 cloud VMs - to process varied task datasets, ranging from 50 to 400 tasks.

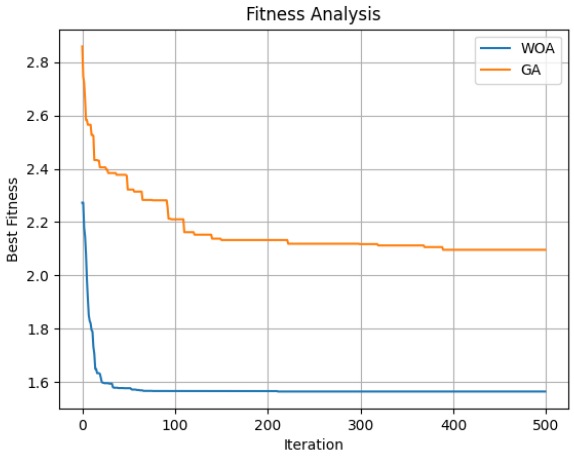

Fig. 4. Fitness Analysis of GWO and GA

The outcome of this experiment is depicted in Fig. 4. We can observe that the GWO algorithm surpasses the GA algorithm in achieving a smaller fitness value. Furthermore, the fitness values for the GWO algorithm demonstrate an increase in line with the growth of dataset sizes, reflecting the characteristics of a minimization problem. This leads us to the conclusion that the GWO algorithm outperforms the GA in every scenario, reflecting its impressive ability to establish a strong balance among the makespan, total execution cost, and energy consumption.

TABLE I
COST, ENERGY, AND MAKESPAN FOR DIFFERENT N VALUES

N	GA Cst	GWO Cst	GA Egy	GWO Egy	GA Mspn	GWO Mspn
100	338.58	320.02	431.06	420.04	14.95	13.03
200	376.29	360.03	471.02	460.01	15.87	14.55
300	414.00	400.01	511.09	500.04	18.90	17.98
400	451.71	440.05	551.15	540.07	16.60	15.30
500	489.42	480.06	591.22	580.09	18.65	15.52

From Table.1, we can see that the GWO has exhibited better performance than the GA in terms of both cost and energy efficiency. GWO shows lower cost values, suggesting it offers a more economical approach to task scheduling. Furthermore, GWO also exhibits lower energy consumption values, meaning that it provides a more energy-saving task scheduling solution. The cost values, expressed in a simulated currency called Grid Dollars (G) [16], signify the amount required to use resources in the computational grid. These values are analogous to the pricing models of various real-world cloud service providers. A lower grid dollar value implies a more cost-effective method of executing tasks on virtual machines. In parallel, the energy consumption values represent the amount of power used in carrying out these tasks on virtual machines. When these values are lower, it signifies that the scheduling solution being employed is more energy efficient. The Makespan values indicate total time needed to complete all tasks. GA tends to have higher makespan values, indicating longer completion times compared to the proposed GWO. This suggests that the GWO method is more efficient, leading to quicker completion of all tasks for all tested value of N.

VI. CONCLUSION

In this research, we presented an innovative approach to a task scheduling problem using a version of GWO that incorporated a semi-dynamic real-time scheduling component. Our strategy targeted this problem as one of minimization, focused on optimizing the balance between total execution cost, energy consumption, and makespan of tasks. The results of our experiments demonstrated the superior performance of GWO in managing task scheduling in a hybrid cloud and fog computing environment, making it an exceptionally efficient solution for real-time, computing problems. Notably, the performance of GWO has been demonstrated, yielding superior results compared to GA across various benchmarks.

VII. LIMITATIONS AND FUTURE WORK

While this study aimed to simulate real cloud-fog environments, using a synthetic dataset within a simulated platform presents certain limitations. The Amazon EC2 framework, despite its computing power, cannot fully capture the intricate complexities of real cloud infrastructures. As such, the direct translation of the experimental findings to practical systems may entail deviations. The random generation of task and VM parameters, however carefully tuned, remains an approximation of real-world attributes. The availability of actual datasets could have enhanced the fidelity of the simulations. Future work could involve validating the results using real-world datasets on physical fog and cloud infrastructures. Testing the proposed approach on systems deployed for commercial or industrial applications could provide greater confidence.

REFERENCES

[1] Shi, Weisong, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. "Edge computing: Vision and challenges." IEEE internet of things journal 3, no. 5 (2016): 637-646.

[2] Mukherjee, Mithun, Lei Shu, and Di Wang. "Survey of fog computing: Fundamental, network applications, and research challenges." IEEE Communications Surveys Tutorials 20, no. 3 (2018): 1826-185

[3] R. O. Aburukba, T. Landolsi and D. Omer, "A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices", J. Netw. Comput. Appl., vol. 180, Apr. 2021.

[4] Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61

[5] T. Ahanger, F. Dahan, U. Tariq, and I. Ullah, "Quantum Inspired Task Optimization for IoT Edge Fog Computing Environment", 1535, Mathematics, vol. 11, no. 1, p. 156, Dec. 2022.

[6] Mujtiba, Syed & Begh, Gh Rasool. (2022). Hybrid Heuristic Algorithm for Cost-efficient QoS Aware Task Scheduling in Fog–Cloud Environment. Journal of Computational Science. 64. 101828.

[7] R. Sing, S. K. Bhoi, N. Panigrahi, K. S. Sahoo, N. Jhanjhi, and M. A. AlZain, "A Whale Optimization Algorithm Based Resource Allocation Scheme for Cloud-Fog Based IoT Applications," Electronics, vol. 11, no. 19, p. 3207, Oct. 2022, doi: 10.3390/electronics11193207.

[8] Memari, Pedram, Seyedeh Samira Mohammadi, Fariborz Jolai, and Reza Tavakkoli-Moghaddam. "A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture",Journal of Supercomputing 78, no. 1 (2022): 93-122.

[9] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo and J. Wang, "DEBTS: Delay Energy Balanced Task Scheduling in Homogeneous Fog 1496 Networks", IEEE IoT Journal, vol. 5, no. 3, pp. 2094-2106, June 2018.

[10] A. Ali et al., "An Efficient Dynamic-Decision Based Task Scheduler for Task Offloading Optimization and Energy Management in Mobile Cloud Computing," Sensors, vol. 21, no. 13, p. 4527, Jul. 2021.

[11] "Virtual Machines and the Internet of Things IoT - Virtual Machine Lab," Virtual Machine Lab, Jan. 08, 2023. [Online]. Available: https://virtualmachinelab.com/?p=1468

[12] Bu, T., Huang, Z., Zhang, K. et al. Task scheduling in the internet of things: challenges, solutions, and future trends. Cluster Comput (2023).

[13] Reddy VD, Nilavan K, Gangadharan G, Fiore U. 2021. Forecasting energy consumption using deep echo state networks optimized with genetic algorithm. In: Artificial intelligence, machine learning, and data science technologies. CRC Press, 205–217.

[14] Beloglazov A, Buyya R. 2012. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual ma-chines in cloud data centers. Concurrency and Computation: Practice and Experience24(13):1397–1420.

[15] K. Alatoun, K. Matrouk, M. A. Mohammed, J. Nedoma, R. Martinek, and P. Zmij, "A Novel Low-Latency and Energy-Efficient Task Scheduling Framework for Internet of Medical Things in an Edge Fog Cloud System," Sensors, vol. 22, no. 14, p. 5327, Jul. 2022

[16] Abohamama, A. S., El-Ghamry, A., & Hamouda, E. (2022). Real-time task scheduling algorithm for iot-based applications in the cloud–fog environment. Journal of Network and Systems Management, 30(4), 54.