Advancing Task Scheduling in Edge Computing for Energy Efficiency: A Multi-Objective Method

Anila Gogineni

Independent Researcher
USA
anila.ssn@gmail.com

Abstract
The rising popularity of the Internet of Things has made energy efficiency an essential factor throughout IoT service system design and development processes. The edge computing model currently receives widespread attention as a modern approach to computing. The presented paper details a novel Task Scheduling Algorithm (TSA), which uses VM heterogeneity to dynamically place tasks based on system requirements. TSA prioritizes execution efficiency using key performance metrics, including task completion time, memory usage, energy consumption, as well as the total running time that it implements in the CloudSim 3.0.3 simulation framework. TSA's superiority is shown to be in comparison with traditional methods of scheduling like First-Come, First-Served (FCFS) and Particle Swarm Optimization (PSO). Experimental data indicates that TSA finishes tasks in 6.6 units, which exceeds both FCFS (69.9 units) and PSO (9.63 units). Additionally, TSA optimizes energy consumption at 3690 units while maintaining efficient memory usage and execution speed. TSA demonstrates its capability to enhance cloud resource control as well as workload placement between multiple nodes which results in an energy-efficient scheduling system.

Keywords: Edge Computing, Task Scheduling, Multi-Objective Optimization, Energy Efficiency, Resource Allocation, Computation Offloading, Low-Latency Computing, IoT, Cloud-Edge Collaboration, Performance Optimization

I. INTRODUCTION

Technology development creates massive amounts of data for processing faster than ever before. Rising technological applications need artificial systems to process data in real time at low delay levels. The high demands of IoT devices on cloud systems create problems because traditional cloud setups have performance challenges caused by heavy network traffic and slow internet connections. The revolutionary introduction of edge computing has allowed for minimal latency and great efficiency by bringing cloud computing to the network's periphery [1]. Real time processing becomes less centralized cloud based and there is improvement in bandwidth utilization and reduced latency [2]. Nevertheless, computational task scheduling and resource allocation in edge environments are most difficult due to the efficiency and throughput of computational tasks [3]. These constraints, coupled with the rationale for load sharing and delaying the amount of processing at the edge device, which is characterized as limited, enhance the need for advancement the advanced scheduling mechanisms[4].

Within this context, one of the greatest challenges is energy efficient task scheduling relating to executing computational tasks in the most energy efficient manner possible while keeping system performance levels in mind [5]. These plans imbalance between time and resource utilization and this has been a general trend among scheduling strategies that optimize either speed or power at the cost of system stability and longevity [6]. These problems are solved by the so-called multi-objective optimization models which optimize one certain goal in several directions at once with respect to such characteristics as time needed to do the particular activity, energy use, and workload allocation.

Thus, multi-objective optimization-based task scheduling stands out as an effective approach to improving the performance of edge computing systems [7]. These models provide optimal ways of decision making and efficient resource allocation in order to optimize the available computing resources without compromising on performance and power consumption. This paper discusses different approaches to applying energy efficiency in task scheduling in edge computing systems where various tasks are allocated based on dynamic aspects of the system. The proposed work will focus on the robustness and scalability of the infrastructure for supporting edge computing so as to meet the new age applications increasing needs.

The contribution of the study

The increase of the edge computing demand for executive functions also triggers the difficulty of effectively scheduling jobs due to their complexity and the requirement to reduce energy usage. These traditional scheduling heuristics do not incorporate a way to achieve the best balance among several objectives like energy, time, and resources. Furthermore, edge computing environments are dynamic in nature, and therefore, it require mechanisms that can change with the dynamic workloads while meeting the set QoS standards. The purpose of this work is relevant to the requirement for an intelligent scheduling solution that can consider both energy consumption and computational cost in edge computing systems. The following contributions of this paper are listed below:

- Task scheduling in edge computing may be made more energy efficient with utilizing an optimization model with many objectives.
- The algorithm is implemented and tested using CloudSim 3.0.3, simulating a cloud environment with multiple Virtual Machines (VMs).
- Tasks are categorized based on computational requirements, and TSA dynamically assigns them to VMs for optimal execution.
- Execution time, energy consumption, and memory utilization are monitored to evaluate the efficiency of TSA.
- Results are analyzed to assess scalability and adaptability in cloud computing scenarios.

Paper Organization

Here is the structure of the paper: The current literature on task scheduling for edge computing is presented in Section II, while the research strategy and methods are detailed in Section III. Section IV covers the analysis of Section V provides the findings and a conclusion with recommendations for further investigation.

II. Literature Review

This literature review discusses several research papers that address multi-task scheduling in edge computing. Table I provides a comparative analysis of optimization techniques in robotics, edge computing, and energy efficiency, summarizing objectives, methodologies, key findings, and limitations.

Singh et al. (2022) developed a multi-objective reinforcement learning technique (FI-MORL) based on fuzzy inference systems to determine the optimal gait parameters for snake-like MRs using the objective weights. In comparison to single-objective reinforcement learning algorithms, FI-MORL outperforms them by 2% in power consumption and 2.5% in velocity gain due to its mitigation of the effect of weight change. This performance is comparable to that of an actor-critic algorithm. Additionally, when compared to more conventional approaches such as simple policy gradient, deep Q-network, and proximal policy optimization, the suggested strategy achieves 11% greater velocity while using 14% less power. After accounting for weight changes, FI-MORL still outperformed the other approaches by 14%. Convergence is achieved more quickly and changes in goal weights are handled efficiently by the suggested FI-MORL framework [8].

Zhang et al. (2022) create ENTS is the first edge-native task scheduling system designed to manage distributed edge resources and enable efficient task scheduling, hence improving the performance of edge-native applications. As compared to the previous methods, entropy-based neural Tigers achieve an average work throughput that is 43 % – 220 % higher, the results of the extensive tests conducted on a real-world edge video analytics scenario [9].

Liu, Cheng, and Wang (2020) examine the four optimization goals for the data center and the VC: energy consumption, availability, average resource use, and resource load balance. In order to accomplish all four optimization goals at once, they next introduce a multi-objective optimization model and discuss an evolution technique. Lastly, test results demonstrate the created algorithm's efficacy and efficiency [10].

Fang, Chen, and Lu (2020) concentrate on reducing the system's power use in order to provide a productive scheduling method for mobile edge computing. A scheduling plan must be supplied to distribute virtual node resources evenly for power efficiency alongside user latency requirements. They simulate their approach using iFogSim. The outcomes of the simulation demonstrate that their approach can successfully lower the edge system's power usage. The greatest energy usage in the idle task test was 27.9% less than that of the original algorithm [11].

Mtshali et al. (2019) propose a virtualization-based application scheduling method for optimizing the energy usage and average latency of apps operating in real time within fog computing networks. The FCFS scheduling strategy leads to improved system metrics based on iFogSim simulations because it reduced energy usage by 11% while lowering task delays by 7.78% and network dependency by 4.4% with a 15.1% shorter execution period [12].

Lu, Dai, and Sun (2019) A set of optimal speed curves for energy conservation and the corresponding energy consumption were generated using the multi-objective particle swarm optimization approach (MOPSO). The gradient descent approach was used to optimize each interstation's trip time allotment. Lastly, the suggested optimization model was tested and validated using actual data from the Yizhuang Line, Beijing Subway. The simulation findings demonstrate that optimizing the bottom layers significantly increases the operation's energy efficiency. Lowering energy usage by 10.7 percent is achieved in the top layer by optimizing the distribution of journey time between stations [13].

Zhang et al. (2018) To reduce the cost of the edge computing system, examine the problem of work schedules. Afterward, they establish that the aforementioned optimization issue is NP-hard. By comparing their method to optimum solutions, they can confirm its efficacy. The findings demonstrate that for the vast majority of the data sets they use, the estimated ratio is lower than 1.2. Results from evaluating their algorithm's performance demonstrate that it successfully meets the latency requirements of all jobs while reducing The price of the system for edge computing [14].

Existing research on multi-task scheduling in edge computing has made notable progress in optimizing individual objectives such as energy efficiency, execution time, and resource utilization. However, a significant gap remains in developing adaptive multi-objective scheduling frameworks that can dynamically balance these conflicting objectives under varying workload conditions. Additionally, scalability issues persist, as many optimization techniques struggle with increased task complexity and distributed resource constraints. In order to overcome these constraints, this research suggests a clever, flexible multi-objective scheduling system that combines evolutionary optimization with reinforcement learning.

TABLE I. SUMMARY OF LITERATURE REVIEW ON OPTIMIZATION TECHNIQUES IN EDGE COMPUTING AND ENERGY EFFICIENCY

Author	Objective	Methodology	Key Findings	Limitations
Singh et al. (2022)	Use multi-objective reinforcement learning to find the best gait parameters for snake-like mobile robots (MRs).	FI-MORL algorithm with fuzzy inference system to accelerate learning and adapt to weight changes	FI-MORL achieved 2% lower power consumption, 2.5% higher velocity than single-objective RL; outperformed traditional methods by 14% lower power and 11% higher velocity	Limited to gait optimization; performance in other robotic applications remains unexplored
Zhang et al. (2022)	Optimize task scheduling for edge-native applications	Kubernetes was extended to create the ENTS system, which employed combined task allocation and flow scheduling problem formulation.	ENTS to the most sophisticated edge computing methods, work throughput increased by 43–220 percent.	Scalability to large-scale heterogeneous edge environments not tested
Liu, Cheng, and Wang (2020)	Optimize load balancing, energy use, availability, and resource use in data centers.	suggested an evolutionary algorithm and a multi-objective optimization model	Experimental results demonstrated effectiveness in balancing multiple objectives	Does not consider real-world system constraints and dynamic workload variations
Fang, Chen, and Lu (2020)	Minimize power use in mobile edge computing while adhering to latency and resource limitations.	Virtualized edge nodes into a master-slave setup; used iFogSim simulation to create a scheduling plan	Achieved up to 27.9% lower energy consumption in idle task scenarios	Assumes ideal network conditions; real-world feasibility remains unverified
Mtshali et al. (2019)	Minimize Fog computing network power use and processing delays	Four job scheduling rules were implemented in a Fog node scheduler; iFogSim simulations were used	Energy use decreased by 11%, task delay by 7.78%, network use by 4.4%, and execution time by 15.1% as a result of the FCFS scheduling mechanism.	Focuses only on FCFS policy; lacks comparison with more advanced scheduling techniques

Lu, Dai, and Sun (2019)	Optimize energy efficiency in subway transportation	Applied MOPSO for energy-saving speed curves; optimized interstation travel time using gradient descent	Improved operational energy efficiency; reduced energy consumption by 10.7%	Limited to subway systems; generalizability to other transport networks not examined
Zhang et al. (2018)	Reduce system costs while meeting edge computing job delay criteria.	Created the TTSCO algorithm and formulated task scheduling as an NP-hard optimization problem	TTSCO reduced system cost effectively; 95% of test cases had an approximation ratio below 1.2	Computational complexity may be high for large-scale edge networks.

III. METHOD AND MATERIAL

In a cloud computing context, the suggested job Scheduling Algorithm (TSA) is made to maximize job distribution across virtual machines (VMs), guaranteeing short execution times, lower energy usage, and effective resource use. The methodology follows a systematic approach that illustrate in Figure 1, beginning with a simulation setup using CloudSim 3.0.3. The cloud environment is modeled with multiple VMs and heterogeneous cloudlets, each with varying computational demands. The scheduling process is controlled by a cloud broker, which gathers resource information and assigns tasks based on predefined optimization criteria. The algorithm prioritizes task execution by considering key performance metrics like task completion time, energy consumption, memory usage, and running time. The experimental setup involves varying task loads and analyzing performance trends using simulation-based evaluations. Comparative analysis is conducted against baseline scheduling techniques, like First-Come, First-Served (FCFS) and Particle Swarm Optimization (PSO), to assess the efficiency of TSA in handling dynamic workloads.

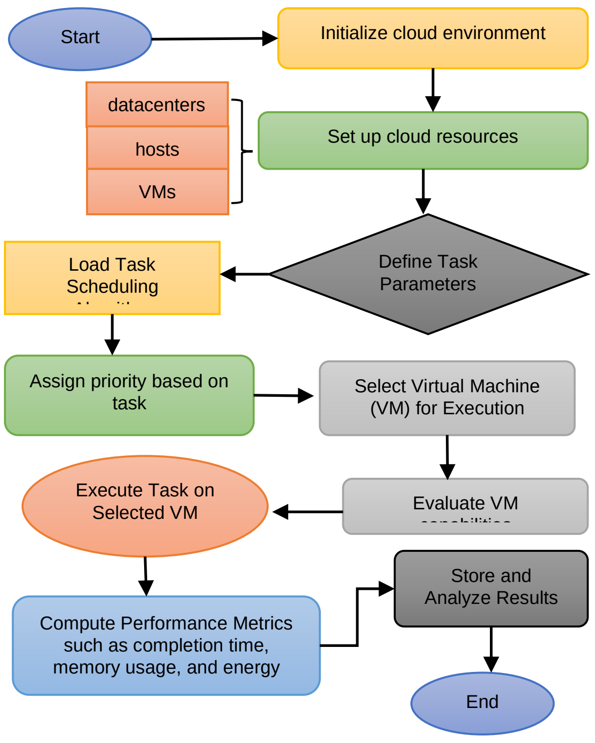
Start
Initialize cloud environment
datacenters
hosts
VMs
Set up cloud resources
Define Task Parameters
Load Task Scheduling
Assign priority based on task
Select Virtual Machine (VM) for Execution
Evaluate VM
Execute Task on Selected VM
Compute Performance Metrics such as completion time, memory usage, and energy
Store and Analyze Results
End

Fig. 1. Flowchart for Task Scheduling for Edge Computing

The overall system implementation process are provided in below:

Task Scheduling Algorithm (TSA)

A Task Scheduling Algorithm is a computational method used to allocate tasks efficiently across available resources in a system while optimizing performance metrics such as execution time, energy consumption, and resource utilization. These algorithms play a critical role in distributed computing, cloud environments, and real-time systems by ensuring tasks are executed in an optimal order to enhance system efficiency[15]. A cloud computing environment's work scheduling is shown in Figure 2 [4].

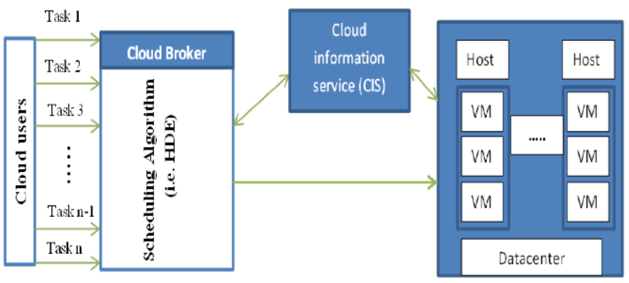

Fig. 2. Task scheduling process in the cloud computing environment [16]

Supposing there are n cloudlets $(Tasks), T = T_1, T_2, T_3, \ldots\ldots, T_n$ $(Tasks), T = T_1, T_2, T_3, \ldots\ldots, T_n$ which are executed using m virtual machines (VMs), $VM = VM_1, VM_2, VM_3, \ldots.., VM_m$.$VM = VM_1, VM_2, VM_3, \ldots.., VM_m$ .These tasks vary in length, and the virtual machines (VMs) have different features including CPU time, RAM, and bandwidth. The cloud broker makes a request to the cloud information service to obtain details about the services needed to finish the tasks that have been assigned. Tasks on the identified services may be scheduled after the information is acquired. The broker's quality of service standards and several other factors influence the choice of projects to be delivered [17]. The cloud broker plays a crucial role in task scheduling by acting as a mediator among the user and the provider and by determining the timestamps for certain resources. Solving an optimization problem, where the objective is to reduce the cost and execution time, is comparable to numerically scheduling jobs, or energy usage while still meeting the limitations of the system. Here is a general formula (1):

$$min\ F(T) = \sum_{i=1}^{n} W_i \,.\, C_i\ min\ F(T) = \sum_{i=1}^{n} W_i \,.\, C_i \quad (1)$$

Where:

- $F(T) F(T)$ is the objective function to be optimized (e.g., makespan, energy consumption).
- n represents the total number of tasks.
- $W_i W_i$ is the weight or priority of task i
- $C_i C_i$ is the completion time of task i.

Performance matrix

They have used the measures listed below to gauge how well their algorithm is doing:

Energy Consumption

It is possible to reduce energy consumption via based task scheduling by distributing work to resources that consume the least amount of power while still fulfilling the requirements of the tasks and the limitations of the resources. Equation (2) describes the mathematical expression of the energy consumption of a job scheduled in cloud computing:

$$EC = P * T\ EC = P * T \quad (2)$$

Energy consumption (EC), power consumption rate (P) of the allocated resource (in watts), and time (T) required to complete the operation are all variables to consider.

Energy Consumption

Data transmission and computing are two of the many operations that cause a fog node to use energy. If the i-th job is to be transferred by the j-th edge device at time t, the total energy consumption may be determined by adding the computation energy $E_{comp} E_{comp}$ and the data transmission energy $E_{tran} E_{tran}$, which are defined as follows (3 to 5)[18]:

$$E_{comp}(i,j,t) = \alpha_1 c_i (f_i)^2 \; E_{comp}(i,j,t) = \alpha_1 c_i (f_i)^2 \quad (3)$$

$$E_{comp}(i,j,t) = \alpha_2 s_i / r(t) \; E_{comp}(i,j,t) = \alpha_2 s_i / r(t) \quad (4)$$

$$E_{Total} = E_{comp} + E_{tran} \; E_{Total} = E_{comp} + E_{tran} \quad (5)$$

$\alpha_1 \alpha_1$ and $\alpha_2 \alpha_2$ represent the energy consumption coefficients in computation and transmission, respectively, $f_i f_i$ is the frequency at which mobile devices' CPU clocks, r(t) is the data transfer rate over the network as at time t, and $s_i s_i$ and $c_i c_i$ are the sizes of the content and the amount of calculation, respectively.

Memory Usage

Memory usage refers to the amount of RAM required to store model parameters, intermediate computations, and input/output data during execution. The following Equation (6) represent the usage of memory.

$$MU = \sum_{i=1}^{n} M_i \; MU = \sum_{i=1}^{n} M_i \quad (6)$$

where: $M_i M_i$ is the memory occupied by component i at any given point in execution.

Running Time

Running time measures the total time taken by an algorithm to execute [19]. It includes model training time, inference time, and overall computational overhead. Mathematical Representation as (7):

$$RT = T_{end} - T_{start} \; RT = T_{end} - T_{start} \quad (7)$$

where:

- $T_{start} T_{start}$ *is the timestamp when execution begins.*
- $T_{end} T_{end}$ *is the timestamp when execution ends.*

IV. RESULT ANALYSIS AND DISCUSSION

The experiment environment, like software and hardware requirements, is present in the next subsection. Then, the experiment results of the proposed algorithm with a graphical representation are provided in the 4.2 section.

Simulation Settings

A Windows 10 computer with an i7-8550U CPU operating at 1.80–2.0 GHz (8 Cores), 16 GB of RAM, and the CloudSim 3.0.3 simulator with Java was used to test and implement the algorithms in this study. The parameters used for the simulation may be found in Table II.

TABLE II. SIMULATION SETTINGS FOR CLOUDSIM ENVIRONMENT

Parameter	**Description**
Operating Systems	Windows, Linux
Virtualization	Supports mature virtualization technology
Resource Management	Virtualized data center resources into resource pools
Simulation Platform	CloudSim
Key Features	Modeling and simulation of virtualized cloud environments
Programming Language	Java
Simulation Components	Hosts, Virtual Machines (VMs), Datacenters, Cloudlets, Network Components
Performance matrix	Running Time, Memory Usage, Completion Time, Energy Consumption

Experiment Results

This section examines how well the recommended task scheduling strategy performs by varying the number of tasks and the average processing power per task. Table III below shows the algorithm's performance with respect to various parameters.

TABLE III. PERFORMANCE OF TASK SCHEDULING ALGORITHM IN CLOUDSIM ENVIRONMENT

Performance Parameters	**Task Scheduling Algorithm**
Task Completion Time	6.6
Total Energy Consumption	3690
Memory Usage	82.374
Running Time	28995.2

The performance metrics of a work scheduling algorithm in a CloudSim simulation environment are detailed in Table III. The task completion time is recorded as 6.6 units, indicating the efficiency of task execution. The total energy consumption is measured at 3690 units, reflecting the power usage of the system during the simulation. Memory usage is reported as 82.374 units, showcasing the resource utilization efficiency. The total running time of the simulation is 28,995.2 units, representing the overall execution duration of the task scheduling process. The measured parameters give information about how well the scheduling algorithm uses computational resources and manages available assets.

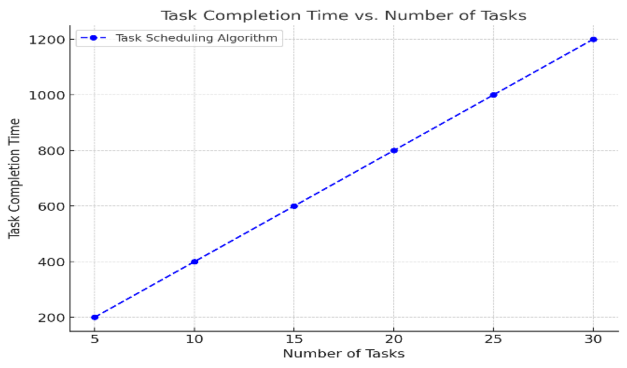

Fig. 3. Line plot for Completion Time

Figure 3 shows a line plot for the Multi-objective Optimization-based Bidding Task Scheduling Algorithm that shows the link between the number of tasks and the task completion time. Task completion time grows from around 200 to 1200 seconds, while the number of tasks range from 5 to 35 seconds on the x-axis. A consistent increasing trend in the figure shows that the task completion time increases as the number of tasks increases.

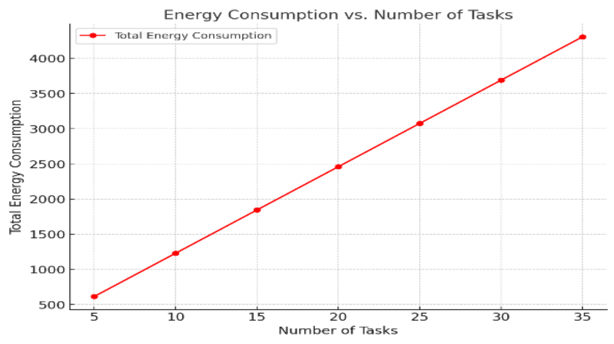

Fig. 4. Line plot for Energy consumption

A line plot showing the correlation between task count and overall power use is shown in Figure 4. The number of jobs is shown on the x-axis, which ranges from 5 to 35, and the total energy consumption is displayed on the y-axis, which goes from about 500 to over 4000. The figure exhibits a steady increasing trend, suggesting that energy consumption increases proportionately with the number of activities.

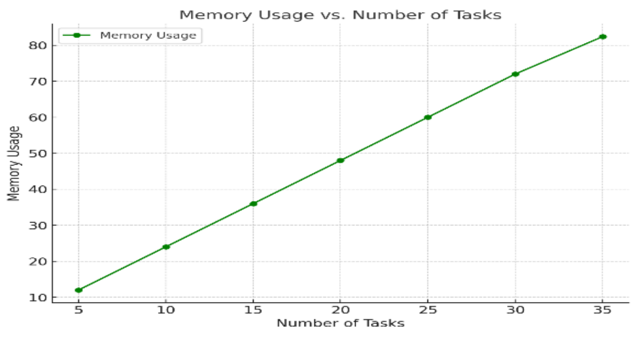

Fig. 5. Line plot for Memory Usage

Figure 5 illustrates a line plot showing a relationship between the number of tasks (x-axis) and memory usage (y-axis), with a strong linear trend. Represented by a solid green line, memory usage increases steadily as the number of tasks rises, starting at around 12 units for 5 tasks and reaching approximately 82 units for 35 tasks. The consistent upward trend suggests a constant rate of memory consumption per additional task, allowing for easy predictions of memory requirements for varying task loads.

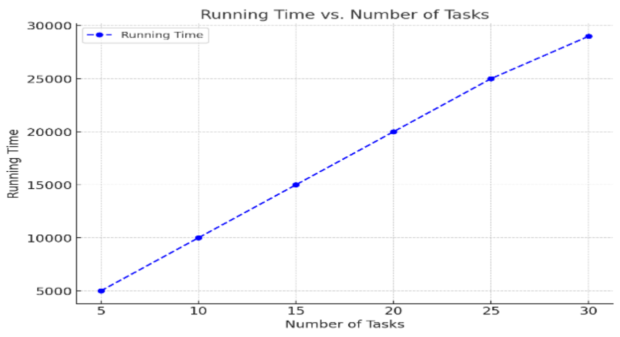

Fig. 6. Line plot for Running Time

The number of jobs (x-axis) and running time (y-axis) are shown in a line plot in Figure 6, which clearly shows a positive link. The running duration steadily grows as the number of tasks increases from five to thirty, beginning at around 5,000 units for five jobs and ending at about 30,000 units for thirty. The relationship appears slightly non-linear, as evidenced by the gentle upward curve of the dotted blue line connecting the data points, which represents the continuous nature of the trend.

Comparison and Discussion

This part compares the multi-objective task scheduling algorithm's performance to that of other algorithms, including First-Come, First-Served (FCFS)[20], and PSO[21]. The task scheduling techniques are evaluated in Table IV in comparison to other current algorithms (like FCFS and PSO) that are based on Matrix.

TABLE IV. COMPARATIVE ANALYSIS OF TASK SCHEDULING ALGORITHMS BASED ON PERFORMANCE METRICS

Measures	FCFS[20]	PSO[21]	TSA
Task Completion Time	69.9	9.63	6.6
Total Energy Consumption	100	-	3690

Table IV presents a comparative analysis of task scheduling algorithms—FCFS, PSO, and TSA. In this comparison, Task completion time varies significantly among the algorithms, with FCFS showing the highest time at 69.9 units, PSO reducing it to 9.63 units, and TSA achieving the lowest at 6.6 units, indicating superior efficiency. In terms of total energy consumption, TSA demonstrates notable optimization with 3690 units, whereas FCFS consumes 100 units, and PSO data is unavailable. This comparison highlights the effectiveness of TSA in minimizing both execution time and energy consumption, making it a more efficient scheduling approach.

V. CONCLUSION AND FUTURE SCOPE

The ability of edge computing to enhance cloud computing performance has led to its widespread application in various contexts. With the feature of distribution in edge computing, computing tasks can be allocated to edge computing systems to reduce the workload of cloud centers. And to compute these tasks in edge nodes can make some aggregation and calculation processes happen close to users. The proposed Task Scheduling Algorithm (TSA) efficiently optimizes task allocation in cloud environments, achieving notable improvements over conventional methods. Compared to FCFS and PSO, TSA significantly reduces task completion time (6.6 units) and energy consumption (3690 units), demonstrating enhanced resource utilization. Simulation results from CloudSim highlight its effectiveness, with memory usage recorded at

82.374 units and a total running time of 28,995.2 units. Despite these advantages, TSA has certain limitations, including potential scheduling delays in highly dynamic workloads and increased computational complexity at scale. Future enhancements should focus on improving adaptability to real-time environments and incorporating predictive machine learning techniques for further optimization.

REFERENCES

[1] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Computing Surveys*. 2019. doi: 10.1145/3362031.

[2] A. and P. Khare, "Cloud Security Challenges : Implementing Best Practices for Secure SaaS Application Development," *Int. J. Curr. Eng. Technol.*, vol. 11, no. 6, pp. 669–676, 2021, doi: https://doi.org/10.14741/ijcet/v.11.6.11.

[3] H. Lu, C. Gu, F. Luo, W. Ding, and X. Liu, "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning," *Futur. Gener. Comput. Syst.*, 2020, doi: 10.1016/j.future.2019.07.019.

[4] B. Boddu, "Cloud DBA Strategies For SQL and Nosql Data Management for Business-Critical Applications," *Int. J. Core Eng. Manag.*, vol. 7, no. 1, 2022.

[5] M. A. Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Syst.*, 2019, doi: 10.1016/j.knosys.2019.01.023.

[6] J. Xue, L. Li, S. Zhao, and L. Jiao, "A study of task scheduling based on differential evolution algorithm in cloud computing," in *Proceedings - 2014 6th International Conference on Computational Intelligence and Communication Networks, CICN 2014*, 2014. doi: 10.1109/CICN.2014.142.

[7] H. K. Langhnoja and P. A. Hetal Joshiyara, "Multi-Objective Based Integrated Task Scheduling in Cloud Computing," in *Proceedings of the 3rd International Conference on Electronics and Communication and Aerospace Technology, ICECA 2019*, 2019. doi: 10.1109/ICECA.2019.8821912.

[8] A. Singh, W. Y. Chiu, S. H. Manoharan, and A. M. Romanov, "Energy-Efficient Gait Optimization of Snake-Like Modular Robots by Using Multiobjective Reinforcement Learning and a Fuzzy Inference System," *IEEE Access*, 2022, doi: 10.1109/ACCESS.2022.3195928.

[9] M. Zhang, J. Cao, L. Yang, L. Zhang, Y. Sahni, and S. Jiang, "ENTS: An Edge-native Task Scheduling System for Collaborative Edge Computing," in *Proceedings - 2022 IEEE/ACM 7th Symposium on Edge Computing, SEC 2022*, 2022. doi: 10.1109/SEC54971.2022.00019.

[10] X. Liu, B. Cheng, and S. Wang, "Availability-Aware and Energy-Efficient Virtual Cluster Allocation Based on Multi-Objective Optimization in Cloud Datacenters," *IEEE Trans. Netw. Serv. Manag.*, 2020, doi: 10.1109/TNSM.2020.2975580.

[11] J. Fang, Y. Chen, and S. Lu, "A scheduling strategy for reduced power consumption in mobile edge computing," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2020*, 2020. doi: 10.1109/INFOCOMWKSHPS50562.2020.9162883.

[12] M. Mtshali, H. Kobo, S. Dlamini, M. Adigun, and P. Mudali, "Multi-objective optimization approach for task scheduling in fog computing," in *icABCD 2019 - 2nd International Conference on Advances in Big Data, Computing and Data Communication Systems*, 2019. doi: 10.1109/ICABCD.2019.8851038.

[13] J. Lu, S. Dai, and X. Sun, "The Application of Multi-objective PSO Algorithm in Energy-efficient

optimization of Metro Systems," in *2019 IEEE 5th International Conference on Computer and Communications, ICCC 2019*, 2019. doi: 10.1109/ICCC47050.2019.9064487.

[14] Y. Zhang, X. Chen, Y. Chen, Z. Li, and J. Huang, "Cost efficient scheduling for delay-sensitive tasks in edge computing system," in *Proceedings - 2018 IEEE International Conference on Services Computing, SCC 2018 - Part of the 2018 IEEE World Congress on Services*, 2018. doi: 10.1109/SCC.2018.00017.

[15] R. Arora, S. Gera, and M. Saxena, "Mitigating Security Risks on Privacy of Sensitive Data used in Cloud-based ERP Applications," in *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2021, pp. 458–463.

[16] H. Ben Alla, S. Ben Alla, A. Touhafi, and A. Ezzati, "A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment," *Cluster Comput.*, 2018, doi: 10.1007/s10586-018-2811-x.

[17] J. Q. Gandhi Krishna, "Implementation Problems Facing Network Function Virtualization and Solutions," *IARIA*, pp. 70–76, 2018.

[18] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and Computation Co-Offloading with Reinforcement Learning in Fog Computing for Industrial Applications," *IEEE Trans. Ind. Informatics*, 2019, doi: 10.1109/TII.2018.2883991.

[19] Prity Choudhary and Vikas Jalan, "Enhancing Process Comprehension through Simulation-Based Learning," *Int. J. Adv. Res. Sci. Commun. Technol.*, pp. 919–924, Dec. 2022, doi: 10.48175/IJARSCT-14400R.

[20] Z. Yin *et al.*, "A Multi-Objective Task Scheduling Strategy for Intelligent Production Line Based on Cloud-Fog Computing," *Sensors*, 2022, doi: 10.3390/s22041555.

[21] W. Li *et al.*, "Multi-Objective Optimization of a Task-Scheduling Algorithm for a Secure Cloud," *Inf.*, 2022, doi: 10.3390/info13020092.