Journal of Theoretical and Applied Information Technology
15th December 2024. Vol.102. No. 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DYNAMIC PROGRAMMING-ENHANCED ENERGY-EFFICIENT TASK SCHEDULING IN EDGE-CLOUD ENVIRONMENTS

DR.SARAVANAN.M.S[1], MRS. MADHAVI KARUMUDI[2]

[1]Professor, Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai.

[2]Research Scholar, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai.
Assistant Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Bowrampet, Hyderabad-500043, Telangana, India.
Email: [1]saravananms@saveetha.com / saranenadu@gmail.com [2]madhavi.karumudi@gmail.com, madhavi.karumudi@klh.edu.in

ABSTRACT

In the wake of modern Internet of Things (IoT) use cases and workflow applications, edge computing plays a crucial role in the faster execution of specific tasks. Therefore, the edge-cloud environment offers computing resources required by modern applications. Using cloud resources for every task may lead to violation of Service Level Agreements (SLAs), as well as causing impediments to the execution of tasks with deadlines. In this research, we suggested a system concept and architecture for effective task scheduling in an edge-cloud setting to address this issue. An algorithm called Dynamic Programming based Energy Efficient Task Scheduling (DPEETS) is introduced for effective work scheduling in edge-cloud environments. Our algorithm exploits dynamic programming, hamming distance termination, and randomization to optimize decisions made in the edge cloud about task scheduling. This suggested algorithm uses a dynamic programming table to keep track of iterations and eliminate unnecessary computations. DPEETS considers multiple objectives such as deadline, energy efficiency, and latency of task execution. The algorithm heuristically converges in a few iterations, leading to energy-efficient scheduling of tasks in the edge cloud. A simulation study has revealed that DPEETS outperforms many existing heuristic algorithms.

Keywords – *Task Scheduling, Cloud Computing, Edge Computing, Dynamic Programming, Energy-Efficient Task Scheduling*

1. INTRODUCTION

The emergence of cloud computing and its associated ecosystem has greatly benefited organizations due to the cost-effective nature of cloud infrastructure and services. With consumers' unprecedented usage of cloud resources globally, it has become crucial for cloud infrastructure to be optimized to leverage performance while adhering to service-level agreements between consumers and service providers. This optimization is essential for achieving an equilibrium satisfying consumers and service providers. With cloud computing, task scheduling is crucial due to the dynamic workloads coming from around the world and is typically handled by various heuristic algorithms [1]. The emergence of edge computing, which provides resources closer to users and helps reduce latency in execution, has led to an increasing trend of using edge and cloud computing resources. Numerous academic researchers have proposed optimal resource utilization strategies based on multiple service-level agreements and global user requirements [2].

Numerous approaches can be used to develop scheduling algorithms, and the literature contains many heuristic algorithms that provide the necessary logic for cloud-based work scheduling. A probable synopsis of the text is as follows: Lu et al. [3] provided a cloud computing pre-migration strategy that maximizes energy efficiency and resource utilization via evolutionary algorithms. Tahir et al. [4] outlined

using evolutionary algorithms (EA) like PSO, ACO, and GA in cloud computing to lower energy consumption. Okba et al. [5] spoke about increasing energy economy, cost-effectiveness, and length while scheduling work on the cloud using NSGA-III. Du et al. [6] proposed a Genetic Algorithm (GA) approach with dynamic objectives (DOGA) for workflow scheduling. Peng et al. [7] suggested parallel scheduling of cloud tasks using a Genetic Algorithm (GA) and MapReduce to shorten the execution duration. Son et al. [8] introduced the TCaS algorithm to schedule tasks in settings using Cloud-Fog, with plans to research and enhance other evolutionary scheduling algorithms. Nevertheless, the requirements of dynamic settings such as cloud computing can be beyond the scope of these heuristic techniques. Additionally, new strategies that may properly employ resources to obtain performance benefits are essential when cloud computing is combined with edge computing and customers wish to leverage edge resources for better latency performance. To make informed scheduling decisions, this study examined a dynamic programming-based approach that maximizes the use of edge and cloud resources through various calculations.

This study presents a system idea and architecture for efficient work scheduling in an edge-cloud environment. The study aims to develop an algorithm for energy-efficient task scheduling in an edge cloud environment. Effective decision-making using dynamic programming is crucial since the edge cloud environment consists of both edge and cloud computing resources. We provide a work scheduling approach for the edge cloud termed Dynamic Programming Energy Efficient Task Scheduling (DPEETS). The approach uses randomization, dynamic programming, and hamming distance termination to enhance job scheduling choices and reduce pointless calculations. DPEETS considers many goals, including energy efficiency, task execution delays, and deadlines. Task scheduling in the edge cloud becomes energy-efficient due to the algorithm's heuristic convergence after a few rounds. Simulation studies show that DPEETS outperforms existing heuristic algorithms. This is the format for the remainder of the paper: Part 2 examines the current approaches to task scheduling in cloud and edge-cloud environments; Part 3 describes the methodology that is suggested, including the system model and algorithm; Part 4 reports the findings of our empirical investigation; Part 5 talks about the limitations of the proposed work; and Part 6 wraps up the investigation and offers recommendations for further research.

2. RELATED WORK

The literature has several different techniques for work scheduling. Velliangiriet al.[1] Outperforming current algorithms such as HPSOGA, GA, ES, and ACO, the proposed Hybrid Electro Search with Genetic Algorithm (HESGA) improves work scheduling. Sindhu et al. [2] put out a bi-objective genetic algorithm that takes application and resource-centric goals into account when scheduling cloud workloads to maximize makes pan and average processor utilization. Lu et al. [3] suggested a pre-migration plan for cloud computing based on evolutionary algorithms, which optimizes CPU, network, and disk dimensions to achieve better resource usage and energy efficiency. Tahir et al. [4] explained how cloud computing can cut energy use using Evolutionary Algorithms (EA), including PSO, ACO, and GA. Okba et al. [5] explained how to utilize NSGA-III to schedule tasks on the cloud while maximizing energy, cost, and duration.

Du et al. [6] discussed scheduling workflow for cloud computing and suggested a Genetic Algorithm (GA) method with dynamic objectives (DOGA) for deadline and cost limitations. Additional evolutionary methods and dynamic, multi-objective features will be investigated in the future. Peng et al. [7] proposed to schedule cloud tasks in parallel using a Genetic Algorithm (GA) and MapReduce to reduce execution time. Future studies might concentrate on real-world graphs and enhance the steps of reduction and communication to achieve even more significant efficiency improvements. Son et al. [8] addressed the problems with fog computing in the Internet of Things by developing the TCaS algorithm for task scheduling in Cloud-Fog scenarios. Other evolutionary scheduling algorithms will be investigated and improved in the future, maximizing time, costs, energy usage, and resources while considering realistic restrictions like deadlines, budgets, and resource constraints. Tao et al. [9] suggested using the Improved Genetic Algorithm (IGA) in open-source IaaS cloud platforms to optimize virtual machine (VM) allocation. Future research will examine additional environmental thresholds and evaluate open-source IaaS cloud systems in real-world settings. Pan et al. [10] explained using the sophisticated Phasmatodea Population Evolution

(APPE) algorithm in a diverse cloud to schedule tasks intelligently. Trials conducted on thirty reference functions yield better results, with quicker convergence and better use of resources.

Liu et al. [11] Tasks involving cloud computing require effective scheduling. The NAGA algorithm suggests OSIG by improving genetics, which optimizes CloudSim resource scheduling. Even with longer execution durations, OSIG performs well enough to justify further study for increased effectiveness. Shojafar et al. [12] enhanced system performance by introducing a hybrid job scheduling method that combines fuzzy theory and genetic algorithms. The new approach dramatically increases efficiency while lowering iteration and considering VM MIPS. Paknejad et al. [13] ch-PICEA-g, a multi-objective algorithm incorporating chaotic systems for better optimization, is suggested to fix the scheduling issue with cloud workflows. Findings demonstrate that it outperforms current algorithms in several performance measures. [14] Hindawi retracts the article due to evidence of systematic manipulation in scope, research description, data availability, citations, and content coherence. Liao et al. [15] aimed to optimize cloud computing resources using a complete approach, focusing on makingpan and energy usage. Subsequent research endeavors will delve into the elements that impact the similarity function, enhance case library procedures, fine-tune algorithm parameters, and expand the applications' reach to service-oriented manufacturing systems.

Duan et al. [16] reduced makespan in CloudSim by introducing an Adaptive Incremental Genetic Algorithm (AIGA) for cloudwork scheduling that is viable and efficient. Compared to other algorithms, AIGA performs better in makespan and computation time. Subsequent research will entail an increasingly intricate mathematical framework and assessment for additional objective functions. Kumar et al. [17] presented an effective method for allocating cloud tasks using the Hybrid Genetic Algorithm–Ant Colony Optimization (HGA–ACO) that takes response into consideration throughput, time, and finish time. The findings of the experiment show an excellent performance. Aziza et al. [18] recommended using a bi-objective evolutionary algorithm to solve the cloud computing problem of NP-hard job scheduling. Future research will examine workflow dependencies, broader application of policies, real-world cloud infrastructure trials, and more optimization factors. Alsadie et al. [19] explained MDVMA, a metaheuristic system for cloud computing that uses dynamic virtual machine allocation and optimal job scheduling. Optimal workflow scheduling strategies are part of the work to come. Jena [20] discussed cloud computing, emphasizing energy and makespan optimization using the TSCSA. Further research should look into more crucial goals and reliable algorithms.

Gad et al. [21] Rapid access to shared resources is provided by the dynamic paradigm of cloud computing. Efficient cloud performance requires careful task scheduling. Forthcoming trends and problems in cloud task scheduling inform future research endeavors. Alaei et al. [22] presented the Improved Differential Evolution (IDE) technique for cloud computing that is fault-tolerant. The suggested method improves scheduling performance and fault tolerance using ANFIS prediction for resource load control and reactive fault tolerance. Subsequent research endeavors to integrate the proposed methodology, carry out practical cloud testing, and assess substitute forecasting methods for more improvements. Sahoo et al. [23] look into combining bacterial and genetic foraging algorithms to create a hybrid job scheduling system for the cloud. The algorithm displays superior performance in convergence, stability, and solution variety, which also minimizes energy consumption and maximizes makespan. Xia et al. [24] responded to the growing need for an effective cloud computing work schedule. The future focus will be exploring the boundaries of binary coding for optimal work efficiency. Saeedi et al.[25] introduces I_MaOPSO, a better many-objective particle swarm optimization technique for cloud computing scientific workflow scheduling. Compared to current algorithms, it performs better and tackles four objectives simultaneously.

Shi et al. [26] examined the efficient ways in which 5-M challenges in challenging continuous optimization issues can be handled by evolutionary computation (EC) techniques. It offers a taxonomy and recommendations for future lines of inquiry to improve EC applications. Sun et al. [28] focused on reducing task completion time while introducing a hybrid PSO_PGA algorithm for cloud job scheduling. The algorithm shows practical efficacy, which improves convergence accuracy and solution quality. Valli et al. [29] introduced GEC-DRP, a genetic algorithm for dynamic resource

scheduling in cloud infrastructure. Through simulation tests, it anticipates virtual machine requirements, improves cost optimization, and confirms efficiency. Khan et al. [30] provided the adaptive genetic algorithm (AGA) and the greedy algorithm for cloud-based real-time work scheduling. AGA performs better than the greedy algorithm in terms of solution quality. According to published research, edge-cloud settings require novel techniques for managing jobs.

2.1 Problem Statement

The growing demand for Internet of Things (IoT) applications and complex workflows necessitates effective task scheduling within edge-cloud environments to meet performance requirements. Relying solely on cloud resources can lead to increased energy consumption and potential Service Level Agreement (SLA) breaches, especially for tasks with strict deadlines. This challenge is compounded by optimizing resource usage while balancing multiple objectives, such as energy efficiency, latency, and deadline adherence. Existing task scheduling methods in edge-cloud environments often fall short of achieving a balance between these objectives. Thus, a robust solution is needed to optimize task scheduling, minimize energy consumption, and ensure timely execution.

3. PROPOSED FRAMEWORK

Here, we provide a workable approach to task scheduling in an edge-cloud setting. Our approach is built upon dynamic programming and further improvements. With energy economy as its top priority, this dynamic programming approach provides an orderly answer to the job scheduling problem in edge-cloud settings. We are analyzing the benefits and drawbacks of local, edge, and cloud resources while using less energy. Efficient task scheduling is crucial for edge-cloud situations to reduce energy consumption and optimize resource usage.

3.1 The System Model

One possible solution to this complex challenge would be to use a dynamic programming (DP) approach. Cloud servers and edge devices are used to build the state or the smallest quantity of energy required to plan the actions utilizing the initial step. The energy costs associated with moving work to edge devices, cloud servers, or local processing are included in the equation of the recurrence relationship. Specifically, it considers the lowest energy needed in each of these scenarios, allowing for a comprehensive analysis of the use of resources. Zero professions need zero energy under the most basic circumstances, yet zero-resource vocations result in limitless energy requirements. The DP table is produced by repeatedly iterating over the number of jobs and available resources using the defined recurrence relation. A tracking procedure may reveal the optimal scheduled option and the equipment for each task. By deftly managing the trade-offs between local, edge, and internet resources, this flexible programming method aims to boost system performance and minimize energy usage.

Journal of Theoretical and Applied Information Technology
15th December 2024. Vol.102. No. 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

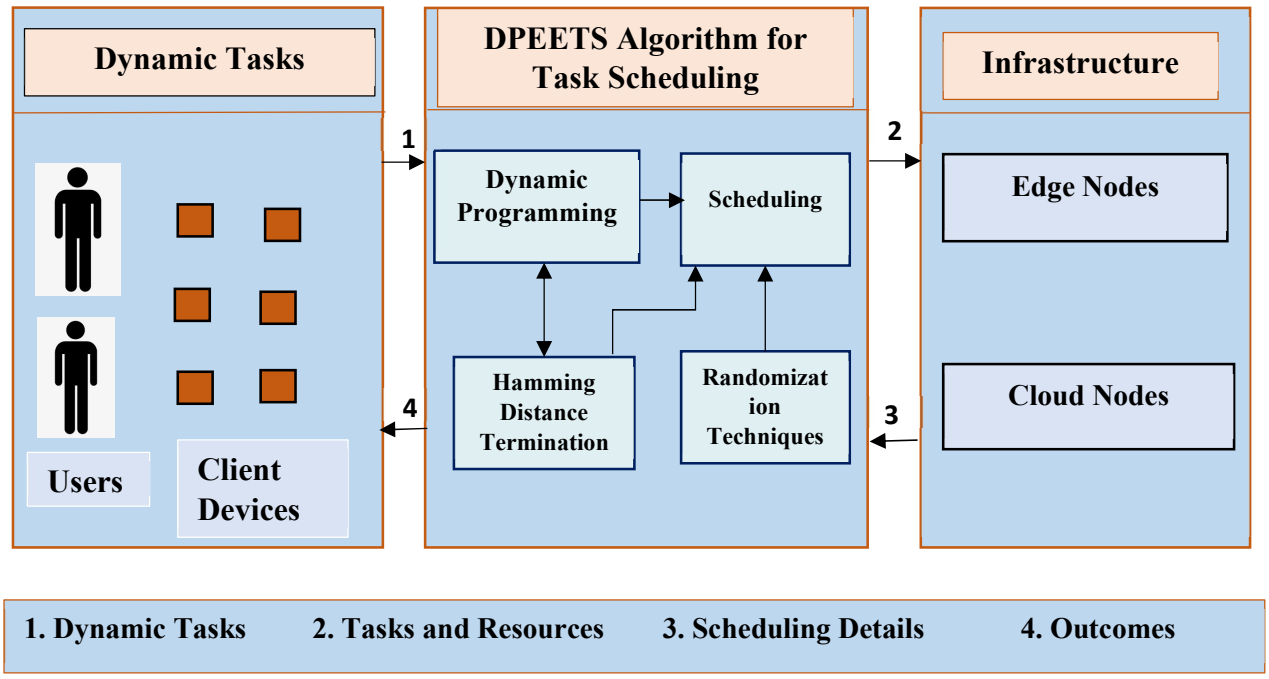

Figure 1: *Proposed architecture for task scheduling in an edge-cloud environment*

The suggested architecture for the task scheduling of an edge-cloud system is shown in Figure 1. It emphasizes how dynamic tasks are routed through a scheduling algorithm (DPEETS) and how they interact with cloud and edge nodes, among other infrastructure components. The three primary parts of the architecture are infrastructure, the DPEETS algorithm for job scheduling, and dynamic tasks. The architecture's section on dynamic tasks represents the source of the functions that require scheduling. Users create dynamic tasks using a variety of client devices. When users engage with these gadgets, they generate activities that call for computational resources, such as memory or processing power, to be used effectively. After that, the jobs are passed to the DPEETS method for scheduling tasks that consider user requests and the most effective way to allocate them to the necessary infrastructure. DPEETS, an algorithm for task scheduling optimization, is the heart of the architecture. Multiple interrelated components make up this algorithm. Assuring that jobs are assigned to increase effectiveness and decrease delay falls within the purview of the sub-module known as dynamic programming. By evaluating the variation (Hamming Distance) between subsequent task allocations, the Hamming Distance Termination feature assists in identifying when the scheduling process has achieved an optimal or nearly optimal state. Randomization techniques prevent local optima and guarantee that the algorithm investigates a broad spectrum of possible solutions. Choosing different scheduling routes for improved results is made more accessible by this.

Once jobs are allocated to particular resources, either at the edge or cloud nodes, based on task needs and current load circumstances, the result of the optimization and randomization procedures is sent to the scheduling sub-module. The infrastructure component represents the computational resources available for the assigned tasks. It comprises tasks requiring low latency, fast response times, or localized processing handled by edge nodes closer to the users. Although nodes at the edge are near the user, they may complete tasks more quickly since they are typically part of a distributed network. Higher-processing-demanding jobs with shorter latency are handled by the cloud nodes. Although being further distant from users, these nodes provide more significant amounts of resources, making them more appropriate for operations that can wait. To give the best scheduling, the DPEETS algorithm manages the dynamic tasks that users (1) define at the start of the engagement. Dynamic programming, termination methods, and unpredictability approaches are used in this strategy to enhance the allocation of these tasks

(2). After that, the work is distributed across the pertinent infrastructure components, such as edge or cloud nodes (3). Finally, the assigned tasks are completed, and the results are returned to the users (4). The design highlights how effective task scheduling is integrated into a hybrid edge-cloud system, guaranteeing an exchange of resources for productivity at work.

3.2 Problem Formulation

To indicate that job i is being executed, let $M_i \in \{0, 1\}$. If job I is completed on the mobile device, set $M_i = 1$; otis used in the subscript base , cap E l , end base , and sub i set $M_i = 0$. The energy us El_iif it is carried out locally. While Er_i indicates the energy consumption of the edge device while job i is done on the cloud, Et_ire reflects the energy-cost of transmitting task i to the cloud server. Task i can be executed remotely on a remote cloud server using the variable Tr_i, whereas task i can be processed locally using the variable Tl_i. Task i's transmission time to the cloud server is represented by the variable Tt_i. The network's transmission bandwidth will determine the amount of transmission energy required to upload each activity. Variations in wireless network capacity will, therefore, impact the choice to offload. For instance, each job's transmission duration would be calculated by dividing the size by the network's transmission rate. Thus, any variation in the transmission rate would affect the final decision to assign this work to a third party. Equations 1 and 2 determine the energy consumption function and execution time.

$$E = \sum_{i \in N}(M_i El_i + (1 - M_i)Er_i + (1 - M_i)Et_i) \quad (1)$$

$$T = \sum_{i \in N}(M_i Tl_i + (1 - M_i)Tr_i + (1 - M_i)Tt_i) \quad (2)$$

With $T_{constraint}$ being the required execution time, all tasks' execution times T must meet the given condition.

$$T < T_{constraint} \quad (3)$$

To keep things simple, we will refer to a vector of binary offloading choices as $M = [M_1, M_2, .., M_N]$. The following is the issue that we are trying to resolve:

$$\min_{Subject\ to:\ T<T_{constraint}} E \quad (4)$$

There are an exponential number of binary value possibilities when there are more jobs. M_i that may be used to find the best answer. Identifying which jobs to transfer to the cloud server to minimize energy usage and fulfill the task's time constraint for completion.

3.3 Algorithm Design

Our algorithm, Dynamic Programming based Energy Efficient Task Scheduling (DPEETS), is suggested as an effective way to schedule tasks in edge clouds. Our algorithm exploits dynamic programming, hamming distance termination, and randomization to optimize task scheduling decisions in the edge cloud. The proposed algorithm uses a dynamic programming table to keep track of iterations and eliminate unnecessary computations. DPEETS considers multiple objectives such as deadline, energy efficiency, and latency of task execution. The algorithm heuristically converges in a few iterations, leading to energy-effective task scheduling in edge cloud environments. Using this method, an N*N table holds the bit-streams that indicate which jobs should be offloaded, where N is the number of tasks. In the first stage, an initial solution is found by generating a random bit stream. The stream is mapped to the table so the next horizontal cell is allocated to 1s, while the following vertical cell is assigned to 0s. The beginning cell is (1, 2) if the stream's initial bit is 1, and (2, 1) if the stream's first bit is 0. Using this method will save additional calculations for typical bit strings. Table 1 lists the notations that are used in the suggested technique.

Table 1: *Notations used in the proposed methodology*

Notation	Meaning
$M_i \in \{0, 1\}$	Variable reflecting execution indicator
i	Given task
El_i, Er_i	Energy consumed by edge device
Tl_i	Execution time in edge
Tr_i	Execution time in the cloud
Tt_i	Denotes transmission time
$T_{constraint}$	Required execution time
T	Execution time

Figure 2 shows an 8 by 8 2D table. For clarification purposes, assume that N = 8 and that the first random stream consists of either 00110110 (red numbers) or 11100110 (black numbers)—two examples are shown. Assume that in every instance, 11000111 is the second random bit stream. Since the initial bit is 1, the beginning (1, 2) is the second stream's cell. The green stream

that results from filling the table according to the previously described guidelines is seen in Figure 2.

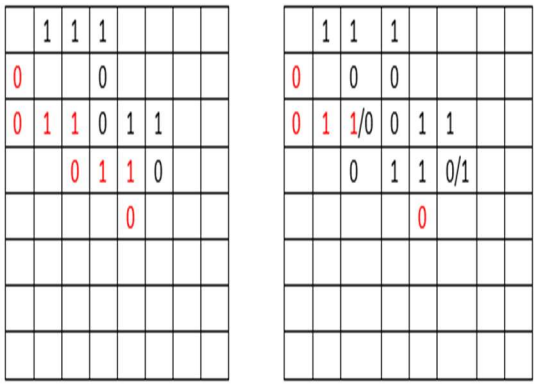

Figure 2: *Dynamically filling tables (left and right)*

Every time a bit stream is randomly formed, we calculate the energy and execution time of every cell or task in the table. We also simultaneously calculate the execution time and total energy consumption of the bit stream. We need to compute the total energy of the new string up to that point to compare it with the existing total energy at the first standard cell if a random bit stream comprising some standard cells with an existing string in the table is produced. The old sub-string is eliminated, the new sub-string is retained, and new quantities are used instead of the cell's old quantities if the new total energy at that specific cell is less than the old total energy. We record the stream's layout in the database each time a new one is created. We stop and accept a solution from an all-one stream whose Hamming distance exceeds a specified threshold. When every component is run locally, it is indicated by the whole stream. In the event where K=20 rounds or 70% of the jobs have been offloaded, for example, the algorithm may terminate. By applying the aforementioned terminating criteria, conclusions are acceptable.

Algorithm: Dynamic Programming based Energy Efficient Task Scheduling (DPEETS)
Inputs: Tasks in descending order T={t_1, t_2, …, t_k}, edge resources C_{edge}={c_1, c_2, …, c_m}, cloud resources VM_{pub}={vm_1, vm_2, …, vm_n}
Output: Efficient scheduling of tasks

1. Begin
2. Initialize edge resources $C_{available}$
3. For each task in T
4. Calculate execution time in edge resources (Tl_i)
5. Calculate execution time in cloud resources (Tr_i)
6. IF $Tr_i < Tl_i$ Then
7. Determine remote execution
8. Else
9. Determine edge execution
10. End If
11. End For
12. For each task t in T
13. Calculate the public cloud cost
14. Calculate edge cost
15. Find priority level of t
16. End For
17. For each task in t in T (priority in descending order)
18. For each resource r in C_{edge}
19. estExeTime←UseRuntimeEstimator(r, t)
20. IF estExeTime<=task deadline Then
21. Update $C_{available}$ with r
22. End If
23. End For
24. For each task t in T
25. IF $C_{available}$ is empty Then
26. Schedule task t in public cloud
27. Else
28. Schedule task t in edge cloud
29. End If
30. End For
31. End

Algorithm: Dynamic Programming based Energy Efficient Task Scheduling (DPEETS)

Algorithm 1 optimizes job scheduling across edge and cloud resources, focusing on energy conservation and meeting work deadlines. The first set of parameters that the algorithm receives is a set of descending-ranked tasks, a set of public cloud virtual machines (VMs), and the presence of edge resources. The final objective is determining the most efficient method for each activity while considering its cost, time, and resource availability. The variable {Cavailable} is created at the start of the process. It records the available edge resources and may be assigned to tasks. This establishes the foundation for subsequent choices on the sites of every activity. To decrease execution time as well as resource usage while maximizing energy efficiency, the

method computes two crucial metrics for each job on the list: on-edge resources ({Tli}) and cloud resources ({Tri}) will be used for the task execution timeframes. At this crucial point, the program decides whether an execution environment is suitable, favoring edge execution over distant execution when the execution time on edge resources ({Tl_i}) is longer than that on the public cloud ({Tr_i}).

The program determines how much it will cost to use edge and cloud resources for activities after deciding on the optimal execution strategy. This involves deducting any running costs for edge devices and fees associated with cloud resource utilization. Duties are ranked in order of priority after the cost analysis. The subsequent scheduling process depends on this priority since it determines how activities are assessed for resource allocation. Priority levels that were previously established provide the basis for scheduling. The decreasing order of relevance of the algorithm means that tasks with higher importance are finished first. Examining each edge resource, the technique estimates the runtime required to complete the current task using a runtime estimator function ({UseRuntimeEstimator}). It indicates that the resource is available for usage if the task is completed within the estimated time of accomplishment.

The process then determines if the inner loop's edge resources are accessible. The job will execute on the public cloud if {Cavailable} has insufficient resources. If edge resources are available, the job is scheduled for edge execution. With this dual-path scheduling approach, you can ensure that work is performed efficiently while meeting deadlines and saving costs. The DPEETS algorithm offers a complete framework for dynamic, energy-efficient job scheduling at the edge and in cloud environments. By integrating execution time estimates, cost evaluations, and priority determinations, the approach deftly navigates the difficulties of resource allocation. This eventually leads to the most cost-effective way to complete tasks in distributed computing systems, improving performance.

4. EXPERIMENTAL RESULTS

In this part, the results of our empirical study are presented. The recommended work scheduling strategy and underlying algorithm were simulated using MATLAB programming. The suggested methodology was compared with earlier approaches when the results were observed. The research's empirical findings might shed light on the usefulness of dynamic programming in cutting-edge cloud computing. Edge cloud computing aims to use resources from edge and cloud computing environments. Improved performance might come from using edge resources.

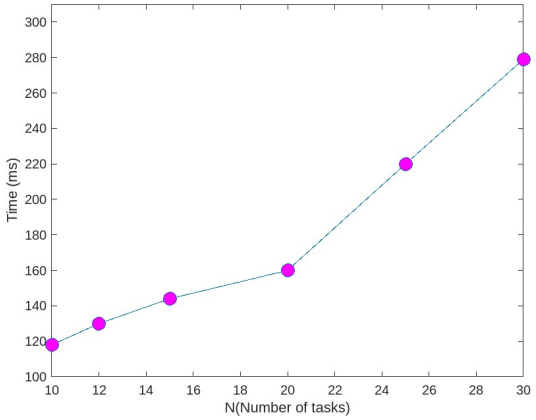

Figure 3: *Results of the proposed Dynamic Programming based Energy Efficient Task Scheduling (DPEETS) algorithm in terms of time against number of tasks*

Figure 3 shows the performance of the DPEETS technique by plotting the time in milliseconds (ms) and the number of jobs (N) on the y- and x-axes, respectively. When there are more tasks—from 10 to 30—the approach takes longer, gradually increasing from 120 ms to about 280 ms. This exhibits the temporal complexity of the DPEETS algorithm concerning task scheduling, showing that its processing time increases as the workload increases.

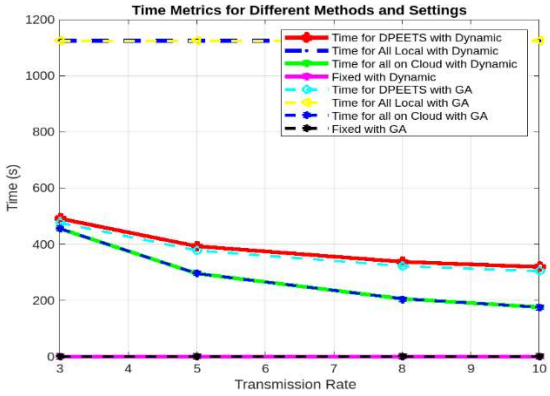

Figure 4: *Results of the proposed Dynamic Programming based Energy Efficient Task Scheduling (DPEETS) algorithm measured in terms of time relative to the state-of-the-art*

Journal of Theoretical and Applied Information Technology
15th December 2024. Vol.102. No. 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 4 compares the time metrics for different methods and settings of task scheduling, including the proposed DPEETS algorithm and a Genetic Algorithm (GA). The transmission rate, which ranges from 3 to 10, is represented by the x-axis, and the duration, which ranges from 0 to 1200 seconds, is shown on the y-axis. Multiple methods are compared: DPEETS with dynamic and GA settings, as well as fixed scheduling with dynamic and GA settings for local, cloud, and all tasks. The graph shows that DPEETS with dynamic scheduling consistently perform better than other methods, with time decreasing as transmission rates increase. The figure caption indicates that the DPEETS algorithm is compared with the most advanced in terms of time efficiency.

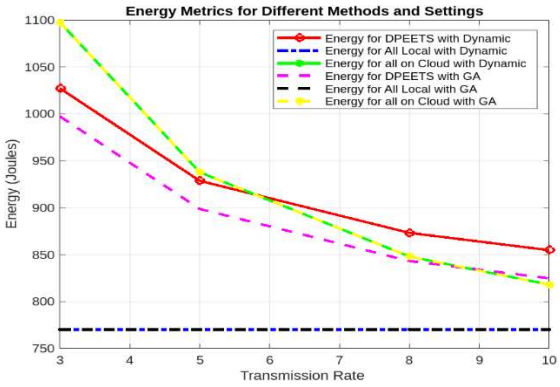

Figure 5: *Results of the proposed Dynamic Programming based Energy Efficient Task Scheduling (DPEETS) algorithm compared to the most advanced in terms of energy usage*

The energy usage of many job scheduling methods at varying transmission speeds is contrasted in Figure 5. The transmission rate is shown on the x-axis, while the energy consumption in joules is shown on the y-axis. The different lines represent different algorithms, including DPEETS with dynamic scheduling, DPEETS with GA, All Local with dynamic scheduling, All Local with GA, All on Cloud with dynamic scheduling, and All on Cloud with GA. The results show that DPEETS with dynamic scheduling consistently exceed the other algorithms' performance in terms of energy consumption across all transmission rates.

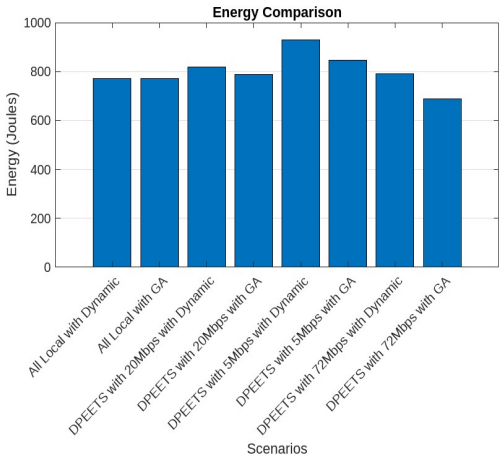

Figure 6: *Results of the proposed Dynamic programming-based Energy Efficient Task Scheduling (DPEETS) algorithm measured in terms of energy efficiency against the state-of-the-art*

The energy usage of many job scheduling methods under various conditions is compared in Figure 6. The y-axis shows the energy consumption in joules, while the x-axis shows the various situations. The different bars represent different algorithms, including All Local with Dynamic, All Local with GA, DPEETS with 20Mbps with Dynamic, DPEETS with 20Mbps with GA, DPEETS with 5Mbps with Dynamic, DPEETS with 5Mbps with GA, DPEETS with 72Mbps with Dynamic, and DPEETS with 72Mbps with GA. Based on all situations, the findings demonstrate that DPEETS with dynamic scheduling consistently use less energy than the other methods, improving performance.

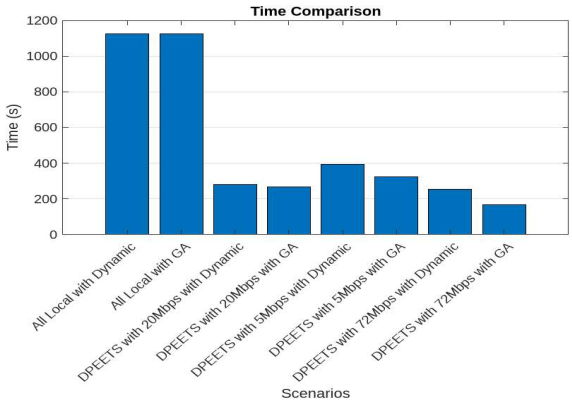

Figure 7: *Results of the proposed Dynamic Programming based Energy Efficient Task Scheduling (DPEETS) algorithm measured in terms of time relative to the state-of-the-art*

Figure 7 evaluates the differences in task scheduling techniques' execution times under

various conditions. The execution duration in seconds is shown on the y-axis, while the various situations are shown on the x-axis. The different bars represent different algorithms, including All Local with Dynamic, All Local with GA, DPEETS with 20Mbps with Dynamic, DPEETS with 20Mbps with GA, DPEETS with 5Mbps with Dynamic, DPEETS with 5Mbps with GA, DPEETS with 72Mbps with Dynamic, and DPEETS with 72Mbps with GA. The results show that DPEETS with dynamic scheduling generally take longer to execute than the other algorithms, especially at higher transmission rates. However, DPEETS with, when it comes to execution time, GA regularly performs better than the other algorithms, especially at lower transmission rates.

5. DISCUSSION

The emergence of edge and fog computing may prove advantageous for enterprises utilizing diverse applications, such as Internet of Things (IoT) systems. The special benefits of cloud computing and edge computing settings bring this about. The several service level agreements between cloud service providers and customers make the latency viewpoint very important. Research indicates that edge cloud, which is closer to the client application, can significantly impact the execution of various applications. Instead of solely relying on cloud computing resources, it is advisable to leverage edge computing resources to enhance the latency performance of different applications. The study found that an edge cloud computing platform can benefit workflow, IoT-based, and other commonly used applications. Throughout this essay, we developed a system model to schedule tasks as they arrive dynamically. The proposed methodology is based on dynamic programming and associated techniques. The research and results indicate that the proposed framework and underlying algorithm could perform reasonably well by utilizing edge resources in conjunction with cloud computing based on task priority and energy efficiency requirements. As mentioned in section 5.1, the suggested approach does, however, have certain drawbacks.

5.1 Limitations

This study suggests several restrictions on the system. First, the study does not use real-time cloud and edge settings; instead, it is based on simulations. Although the simulated environment can provide insightful information, real-time system testing is required to gain discoveries that can be applied to other situations. The simulation's reliance confined to a limited number of tasks and resources, which might limit how widely the outcomes can be used, is another serious flaw in this work. Furthermore, artificial intelligence-enabled methods are not included in the suggested system even though they are crucial for improving performance in our next work.

6. CONCLUSION AND FUTURE WORK

We present a comprehensive system concept and architecture aimed at optimizing the scheduling of tasks within an edge cloud environment—an increasingly important area in cloud computing. Our innovative approach for improving work scheduling decisions in this setting is the Dynamic Programming-based Energy Efficient Task Scheduling (DPEETS) technique. This technique stands out because it integrates several advanced methodologies, including Hamming distance termination, dynamic programming, and randomization, to significantly enhance the decision-making process for scheduling tasks. The incorporation of Hamming distance helps assess the similarity among various tasks, enabling more informed decisions on task grouping and execution order. At the heart of DPEETS is a dynamic programming table, which plays a crucial role in tracking the various iterations of the scheduling process. This table allows the system to minimize redundant calculations, thus streamlining the scheduling process and making it more efficient. Furthermore, DPEETS considers multiple objectives during task scheduling, including energy efficiency, minimizing delays in task execution, and ensuring system reliability. Once the DPEETS algorithm heuristically converges after a few iterations, the edge cloud system can schedule jobs that consume low energy while maintaining optimal performance. This capability is particularly beneficial in environments where energy consumption is critical, such as edge computing scenarios, where resources may be limited. Our simulation studies provide compelling evidence of the effectiveness of DPEETS, demonstrating that it outperforms several existing heuristic techniques commonly used for task scheduling in edge cloud settings. The results indicate that DPEETS enhances energy efficiency and reduces task execution delays, contributing to a more robust cloud computing framework. Looking forward, we intend to explore the implementation of a deep reinforcement learning-based architecture to

advance task scheduling efficiency further. This future direction aims to leverage the strengths of machine learning to create even more adaptive and intelligent scheduling strategies that can dynamically respond to changing conditions and workloads in the edge cloud environment.

REFERENCES

[1] Velliangiri, S.; Karthikeyan, P.; Arul Xavier, V.M. and Baswaraj, D. (2020). Hybrid electro-search with genetic algorithm for task scheduling in cloud computing. Ain Shams Engineering Journal, pp.1–9. doi:10.1016/j.asej.2020.07.003

[2] Sindhu, S. and Mukherjee, Saswati (2013). A genetic algorithm-based scheduler for cloud environment. , IEEE, pp.23–27. doi:10.1109/iccct.2013.6749597

[3] Chen, Shi; Wu, Jie and Lu, Zhihui (2012). A Cloud Computing Resource Scheduling Policy Based on Genetic Algorithm with Multiple Fitness. , IEEE, pp.177–184. doi:10.1109/CIT.2012.56

[4] Maryam, Khola; Sardaraz, Muhammad and Tahir, Muhammed (2018). Evolutionary Algorithms in Cloud Computing from the Perspective of Energy Consumption: A Review, IEEE, pp.1–6. doi:10.1109/ICET.2018.8603582

[5] Latreche Imene a,⇑ , Slatnia Sihem a , Kazar Okba a,b , Batouche Mohamed. (2022). A third generation genetic algorithm NSGAIII for task scheduling in cloud computing. Journal of King Saud University – Computer and Information Sciences, p.7515–7529.

[6] Chen, Zong-Gan; Du, Ke-Jing; Zhan, Zhi-Hui; Zhang, Jun (2015). Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm. , IEEE, pp.708–714. doi:10.1109/CEC.2015.7256960

[7] Zhihao Peng, Poria Pirozmand, Masoumeh Motevalli and Ali Esmaeili. (2022). Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework. Hindawi, pp.1-11.

[8] Nguyen, Binh Minh; Thi Thanh Binh, Huynh; The Anh, Tran and Bao Son, Do (2019). Evolutionary Algorithms to Optimize Task Scheduling Problem for the IoT Based Bag-of-Tasks Application in Cloud–Fog Computing Environment. Applied Sciences, 9(9), pp.1–20. doi:10.3390/app9091730

[9] Zhong, Hai; Tao, Kun and Zhang, Xuejie (2010). An Approach to Optimized Resource Scheduling Algorithm for Open-Source Cloud Systems. , IEEE, pp.124–129. doi:10.1109/ChinaGrid.2010.37

[10] An-Ning Zhang, Shu-Chuan Chu, Pei-Cheng Song, Hui Wang and Jeng-Shyan. (2022). Task Scheduling in Cloud Computing Environment Using Advanced Phasmatodea Population Evolution Algorithms. MDPI, pp.1-16.

[11] Liu, Shaojie and Wang, Ning (2020). Collaborative Optimization Scheduling of Cloud Service Resources Based on Improved Genetic Algorithm. IEEE Access, pp.1–13. doi:10.1109/ACCESS.2020.3016762

[12] Kömer, Pavel; Abraham, Ajith; Snášel, Václav (2014). Hybrid Job Scheduling Algorithm for Cloud Computing Environment, Advances in Intelligent Systems and Computing, pp.43–52. doi:10.1007/978-3-319-08156-4_5

[13] Paknejad, Peyman; Khorsand, Reihaneh and Ramezanpour, Mohammadreza (2020). Chaotic improved PICEA-g-based multi-objective optimization for workflow scheduling in a cloud environment. Future Generation Computer Systems, pp.1–31. doi:10.1016/j.future.2020.11.002

[14] Zhuoyuan Yu. (2023). Retracted: Research on Optimization Strategy of Task Scheduling Software Based on Genetic Algorithm in Cloud Computing E, Hindawi, pp.1-10.

[15] Tao, Fei; Feng, Ying; Zhang, Lin and Liao, T.W. (2014). CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. Applied Soft Computing, 19, pp.264–279. doi:10.1016/j.asoc.2014.01.036

[16] Duan, Kairong; Fong, Simon; Siu, Shirley; Song, Wei and Guan, Steven (2018). Adaptive Incremental Genetic Algorithm for Task Scheduling in Cloud Environments. Symmetry, 10(5), pp.1-13. doi:10.3390/sym10050168

[17] Senthil Kumar, A. M. and Venkatesan, M. (2019). Multi-Objective Task Scheduling Using Hybrid Genetic-Ant Colony Optimization Algorithm in Cloud Environment. Wireless Personal Communications, pp.1–14. doi:10.1007/s11277-019-06360-8

[18] Aziza, Hatem and Krichen, Saoussen (2017). Bi-objective decision support system for task-scheduling based on genetic algorithm in cloud computing. Computing, pp.1–27. doi:10.1007/s00607-017-0566-5

[19] Deafallah Alsadie; (2021). A Metaheuristic Framework for Dynamic Virtual Machine Allocation With Optimized Task Scheduling in Cloud Data Centers . IEEE Access, pp.1–16. doi:10.1109/access.2021.3077901

[20] Jena, R.K. (2017). Energy Efficient Task Scheduling in Cloud Environment. Energy Procedia, 141, pp.222–227. doi:10.1016/j.egypro.2017.11.096

[21] Essam H. Houssein;Ahmed G. Gad;Yaser M. Wazery;Ponnuthurai Nagaratnam Suganthan; (2021). Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends . Swarm and Evolutionary Computation, pp.1-41. doi:10.1016/j.swevo.2021.100841

[22] Alaei, Mani; Khorsand, Reihaneh and Ramezanpour, Mohammadreza (2020). An adaptive fault detector strategy for scientific workflow scheduling based on improved differential evolution algorithm in cloud. Applied Soft Computing, pp.1–16. doi:10.1016/j.asoc.2020.106895

[23] Sobhanayak, Srichandan; Turuk, Ashok Kumar; Sahoo, Bibhudatta (2018). Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. Future Computing and Informatics Journal, pp.1-30. doi:10.1016/j.fcij.2018.03.004

[24] Yiqiu, Fang; Xia, Xiao; Junwei, Ge (2019). Cloud Computing Task Scheduling Algorithm Based On Improved Genetic Algorithm. , IEEE, pp.852–856. doi:10.1109/ITNEC.2019.8728996

[25] Saeedi, Sahar; Khorsand, Reihaneh; Ghandi Bidgoli, Somaye and Ramezanpour, Mohammadreza (2020). Improved Many-objective Particle Swarm Optimization Algorithm for Scientific Workflow Scheduling in Cloud Computing. Computers & Industrial Engineering, pp.1-41. doi:10.1016/j.cie.2020.106649

[26] Zhi-Hui Zhan; Lin Shi; Kay Chen Tan and Jun Zhang; (2021). A survey on evolutionary computation for complex continuous optimization . Artificial Intelligence Review, pp.1–52. doi:10.1007/s10462-021-10042-y

[27] Kaili Shao, Hui Fu and Bo Wang. (2023). An Efficient Combination of Genetic Algorithm and Particle Swarm Optimization for Scheduling Data-Intensive Tasks in Het. MDPI, pp.1-17.

[28] Xueliang Fu; Yang Sun; Haifang Wang and Honghui Li; (2021). Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm . Cluster Computing, pp.1-10. doi:10.1007/s10586-020-03221-z

[29] K. Lalitha Devi and S. Valli; (2021). Multi-objective heuristics algorithm for dynamic resource scheduling in the cloud computing environment . The Journal of Supercomputing, pp.1–29. doi:10.1007/s11227-020-03606-2

[30] Mahmood, Amjad; Khan, Salman and Bahlool, Rashed A. (2017). Hard Real-Time Task Scheduling in Cloud Computing Using an Adaptive Genetic Algorithm. Computers, 6(2), pp.1–21. doi:10.3390/computers6020015