International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(5s) 2026 pp. 332-350
https://doi.org/10.70917/ijcisim-2026-2711
Article

Energy-Efficient Task Scheduling in Fog-Cloud Environments Using an Improved Single Candidate Optimizer

Malik Shahzad Ahmed Iqbal[1,*], Mohammad Haroon[2]

Department of Computer Science and Engineering, Integral University, Lucknow
Email: msai@student.iul.ac.in, mharoon@iul.ac.in
Corresponding Email: msai@student.iul.ac.in

Abstract: Fog-cloud computing has emerged as a promising paradigm for supporting latency-sensitive and data-intensive Internet of Things (IoT) applications through the use of fog computing to bring cloud-style functions closer to the end user. Task management in these heterogeneous environments is challenging due to dynamic workloads and resource constraints; thus, there are also difficulties in balancing the trade-offs among energy consumption, execution time, and operational cost. Metaheuristic-based scheduling algorithms have typically exhibited relatively high computational complexity, slow convergence rates, and poor adaptability to real-time conditions. In this study we present an energy efficient task scheduling framework using an improved single-candidate optimizer (ISC-F). The proposed algorithm improves the existing single-candidate optimizer (SCO) by using a two-phase optimization method that combines exploration and exploitation effectively. In addition, ISC-F allows fog nodes to be prioritized for the execution of latency-sensitive tasks, while dynamically offloading tasks to cloud resources as network conditions change. The optimization process considers multiple objectives including minimum energy consumption, makespan, execution time, and total cost. To evaluate the potential of ISC-F against competing scheduling algorithms, we conducted a series of extensive simulations comparing the aforementioned competing scheduling methods (HEFT, IHEFT, IKH-EFT, IWO-CA). The results demonstrate that ISC-F significantly outperforms existing scheduling techniques; therefore, there is strong evidence that the proposed optimization algorithm provides significant benefits to both fog-cloud computing systems and their respective users.
Keywords: Internet of Things, Single candidate optimiser, Metaheuristic, Cloud-fog computing, Makespan, Energy Efficiency, Task scheduling.

1. INTRODUCTION

The technological marvel known as the Internet of Things (IoT) enables numerous machine-to-machine interfaces and intelligent applications: ecological monitoring and forecasting; intelligent automobile; intelligent metropolitan centers; intelligent buildings; and emergency management Alsharif, M. H. et al (2024), Padmanaban et al (2022). The more the IoT develops, the more ways we'll see our lives transformed as this capability grows exponentially S. K. Dwivedi et al (2022). Currently, it is anticipated that in the not too distant future, this growth will result in a great increase in the number of devices connected to the internet. Furthermore, estimates suggest that the IoT will connect billions of people and their respective devices, providing benefits for everybody Sehgal, N. et al (2023). Each IoT device creates a massive amount of data, which is maintained in the cloud. These connected devices have an extremely high demand for computing resources, therefore their data must be transmitted to cloud-based services for long-term storage, processing, analytic activities, and decision-making. While there are many benefits of using cloud computing. In addition, there are many challenges associated with the use of the cloud for IoT devices: i.e. complexity, rapid advances in cost, lack of capacity utilization; high rate of failure, dependence upon a cloud service provider to successfully execute the aforementioned processes, and the inability to manage delay for real-time control of IoT devices Ramezani Shahidani, F. et al (2023) and Khan, A. M. et al (2015). Because of these challenges, many researchers are looking into the use of fog and edge computing for managing IoTFog computing is what is called

a distributed system used to combine multiple data center computing resources with edge devices connected to the network to serve the various users and applications that require more flexible means of communication, processing, and stored data. Fog computing has the ability to improve the performance of cloud computing by reducing the number of data transfers between devices. Thus, by using connected devices to process, the fog computing model provides significant innovation in how interconnected devices will improve processing capability and improve processing efficiency without displacing existing cloud computing infrastructure. Examples of devices that could be used for the delivery of both services and applications within a fog computing architecture include smart gateways, routers, access points, set-top boxes, roadside units, and vehicle-to-vehicle (V2V) gateways Alizadeh M.R. et al (2020). A fog computing architecture is developed from three tiers being Cloud, IoT, and the supplementing tier of the edge devices. All of the IoT devices generating the data are part of the IoT layer where the raw data will be processed before sending it to either the cloud or fog tier architectures to complete the process. The resource management tier (the second tier) is made up of the routers, edge servers, gateways, switches, and cloudlets and is responsible for managing all of the resources being used by the IoT Application tier by processing, capturing, and returning requests and data. The uppermost layer is composed of both Virtual Machines and Data Centers. This group of clouds also links to the fog-through specific method of connecting to fog. The job scheduling component is also an important part of the cloud and fog designs in terms of how jobs will be executed at the fog, as well as the final user deployment. Efficient task scheduling can help cut down on costs and shorten the time taken for many types of applications such as health care monitoring, smart housing and control of vehicles on the road. The operating system provides a mechanism to manage tasks by establishing methods of assigning resources (CPU, memory, input/output) so that all tasks are completed efficiently within a given amount of time. The OS is a critical part of any OS and increases the performance of the overall system, increases the efficiency of resource use and helps guarantee that all tasks in the operating system will be executed according to the order set by the priority of each. The main objective of scheduling tasks is to assign the job to the appropriate personnel so that the resource assigned to them can complete the job while also meeting the required performance goal for the service. Although there are many advantages of cloud/fog computing, the scheduling of tasks in these types of systems is more difficult because of their dynamic character, the design of the job and the resource demand for the job. Each component of QoS optimization through parameter alterations and the selection of cloud and fog resources affect QoS optimization. According to Varshney, P., & Simmhan, Y (2020) and Chandramani Singh, et al. (2023) task preparation in this environment is non-deterministic polynomial time, which leads to an approximate solution by the random search method. Abohamama, A. S., et al (2022) note that general metaheuristics used for task scheduling include Genetics Algorithms (GA), Ant Colony Optimization (ACO), Simulated Annealing (SA), and Particle Swarm Optimization (PSO) (Saif, F. et al. 2022). Most metaheuristics, particularly in later iterations, have poor convergence characteristics, cause solutions to find locally optimal searches, involve many searches, create random issues, and demonstrate poor global search capabilities (Deng, Z., et al. 2020). Both local and global programs employ radically different methodologies to arrive at either situation (Paknejad, P., et al. 2021). The Grey Wolf Optimiser (GWO) is being analysed by many academia because GWO outmore metaheuristics in many ways. GWO uses a vector with one location to use less space in memory than for the PSO method. GWO does not select one solution and only allows each particle to cluster around one local optimum; GWO explores the top three solutions. Because the GWO uses fewer parameters than competition, it has less complexity, shorter compute time and lower energy consumption than competitors. Faris, Het. However, these studies either consider the challenge of scheduling static operations or ignore real-time job deadlines for IoT applications. Despite their success in small-to-medium computer systems, static scheduling algorithms fail in dynamic environments like the IoT Singh, G., et al (2021) More research is needed to develop employment planning algorithms that can accommodate real-time activities with varying arrival times. The main contribution of this paper is as follows:

1. The suggested approach improves the standard Single Candidate Optimiser (SCO) with a novel two-phase strategy that strengthens exploration and exploitation, culminating in the Improved Single Candidate Optimiser for Cloud-Fog Task Scheduling (ISC-F).
2. ISC-F prioritises fog nodes for low-latency and energy-efficient task execution, optimising resource allocation based on availability and connection, resulting in significant energy savings.
3. The solution efficiently addresses the essential difficulties of energy-efficient scheduling in fog-cloud systems by enabling intelligent task offloading, minimising dependency on energy-intensive cloud data centres, and dispersing workloads to avoid node overload.
4. The proposed ISC-F outpaces existing methods in positions of efficiency and reliability, as measured by key performance characteristics such as cost, makespan, execution time and energy usage.

This is how the remainder of the paper is organised. Related studies on scheduling techniques are covered in Section 2. The objective function, problem formulation, and our suggested solution, ISC-F, which is a superior optimisation for task scheduling in cloud fog, are all explained in Section 3. The results and an evaluation of the suggested strategy are presented in Section 4. Finally, Section 5 includes some concluding remarks, as well as some prospective possibilities for further inquiry.

1.1 PROBLEM STATEMENT:

Although fog-cloud computing provides various advantages in helping with latency-sensitive and data-intensive IoT applications, efficient task scheduling continues to be an important issue. Because of the heterogeneity and distributed nature of the fog and cloud resources along with the dynamic workloads that arrive at different times, along with the wide variations in network conditions, it is difficult to identify and allocate the most optimal resources, as stated by Suhel Ahmad Khan, et al., 2020. Existing scheduling methods based on metaheuristics typically have high computational complexity, convergence times that are slow to achieve, and very limited adaptability for real-time environments. In addition, many existing solutions do not address the simultaneous optimization of several, conflicting objectives (e.g., energy consumption, execution time, and operational cost), resulting in inefficient processing of available resources, high energy consumption, and an increased likelihood of QoS failures. Thus, there is an immediate need for an efficient mechanism for executing tasks in a cloud-fog computing environment that can support dynamic workloads with optimal performance.

1.2 RESEARCH GAP:

While there have been many proposed scheduling methods regarding fog-cloud systems, there are still numerous unsolved issues causing some of these methods to be inadequate. Traditional metaheuristic algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) are typically used in most existing approaches, but all 3 tend to converge slowly and have high computational costs, therefore are not suitable for real-time and large-scale Internet of Things (IoT) applications. Most current methodologies are single-objective focused methodologies (makespan or cost), so there is very little effort/questions on how to jointly optimize for Energy usage, Latency and Resource Utilization. Most existing solutions do not have the ability to adjust to dynamic workload patterns and changing network conditions, leading to sub-optimal scheduling decisions in real-world implementations. Another issue many of the current scheduling methods face is that there is a lack of balance between the exploration & exploitation trade-off during the optimization process, which leads to either exceeding to converge prematurely or search perform poorly. Consequently, there exists an open research opportunity to create a lightweight, fast-converging and multi-objective scheduling framework that is capable of adapting dynamically to variations within fog-cloud environments while at the same time being Energy efficient and providing adequate Quality of Service (QoS).

2. RELATED WORKS

A study by Iftikhar et al. 2023 compared the effects of replacing GGCN's Gated Recurrent Unit (GRU) with a bidirectional GRU. The study analyzed the use of Convolutional Neural Networks (CNNs) for the optimization of cloud-fog scheduling. Using the energy consumption of each completed task as well as the job completion rates as performance metrics, the authors concluded that the CNN scheduler outperformed GGCN-based models by a minimum of 17% in regards to both metrics. The authors cautioned that the increase in complexity associated with processing and what is required by resources will have a detrimental effect on any real-time applications.

In another study conducted by Khaledian et al. 2023, a system for workflow scheduling in Fog and Cloud Architecture was presented, which aims to improve customer satisfaction by reducing costs and increasing energy efficiency. As there are multiple variables involved and the requirement to optimize conflicting targets, this problem is NP-hard. To find an optimal solution to this problem, a krill herd technique was implemented as a multi-objective metaheuristic approach. The initial population of solutions was intentionally designed for rapid convergence. The Energy-efficient Fog Cloud Tasks (EFT) method assigns tasks to available fog cloud resources based on their dynamic voltage and frequency levels to minimize total energy consumption. A simulation was run with multiple scenarios with varying CCR values to evaluate the proposed approach versus other techniques such as the IHEFT, HEFT, and IWO-CA approaches. Overall, the makespan outperformed all other techniques in terms of energy consumption and cost in the fog cloud environment. As with any multi-objective optimization problem, there will be a tradeoff between cost and energy usage due to NP-hardness of balancing two conflicting objectives and resulting computation difficulty and execution time especially for dynamic and/or large-scale scenarios.

Subramoney and colleagues (2023) designed a Multi-Swarm Particle Swarm Optimization (MS-PSO) algorithm that enhances cloud-fog methods for scheduling scientific processes. The MS-PSO algorithm overcomes premature convergence, which is very undesirable and can often lead to suboptimal solutions when using standard PSO approaches by implementing rules to group particles into swarms that utilize success and experience of individual particles to create "social" and "cognitive" types of learning. The FogWorkflowSim Toolkit assesses the weighted total objective function (cost, makespan, energy and load balancing) for process scheduling using the MS-PSO technique for both published and unpublished processes in multiple modes. According to the outcomes of this study, it was determined that MS-PSO had a significantly higher degree of stability and reliability than that of domestic PSO (The second-best performance was in Execution Time) and MS-PSO excelled beyond PSO across all of the scientific measures and metrics. However, the complexity that MS-PSO imposes on the algorithms leads to higher computing costs and slower rates of convergence compared to the algorithms used to solve multiple objectives (i.e., makespan, cost, energy, and load). As a result, the use of MS-PSO may be limited for large-scale or real-time systems.

Khaledian, N. et al. (2024) addressed the challenge by viewing IoT applications as a set of dependent tasks. Applying priority levels and order of execution to jobs performed as a group (i.e., beneath the same process and level of hierarchy) led to a substantially more complex question and one that better aligned with real-world applications. To improve fitness function and establish a priority for performing tasks, a hybrid model known as PSO-SA was introduced. The proposed model optimises makespan and energy consumption at the nodes of the fog-cloud ecosystem, thereby regulating the allocation of jobs. The replication results confirmed that the PSO-SA model outperformed the traditional method (i.e., IKE-EFT) in terms of energy usage and makespan performance. However, in either dynamic or large-scale environments, the PSO-SA could potentially have slower convergence rates and escalate processing complexity. Therefore, fine-tuning or calibrating the parameters for both algorithms may prove to be difficult, thereby adversely affecting the overall performance of the system.

In 2022, J. U. Arshed et al. put forth a genetic algorithm for allocating Cloud-Fog resources by employing a schedule strategy for the optimal mapping of presentation units. This method made use of Execution Time as a fitness function in order to allocate application modules to maximize the usage of the Fog Devices that are accessible. Execution Time, Cost, and Bandwidth values were derived from the proposed method and then compared to those produced by existing scheduling methods referenced by the literature. The replicated results support the performance of this new solution when compared to previous examples of scheduling algorithms referenced in the literature. However, the ability of the algorithm to adapt to changing workloads and to work efficiently in large solution spaces is compounded by such flaws as long convergence times; high processing costs; and problems in changing solution parameters, such as Mutation and Crossover rates.

In 2022, V. Sindhu et al. studied an innovative Resource Allocation and Task-Scheduling Algorithm that provides improvements in Resource Allocation and the Task Scheduling Performance while still minimizing the Cost and Energy Consumption associated with execution of the task. The proposed method makes use of Directed-Acyclic graphs (DAG) as a way to express the relationships of tasks with a Workload Priority assigned to each task in order to establish the relationship associated with each task. The proposed method was able to balance Scheduling Time, Cost, and Energy Utilization in such a manner that it could achieve the maximum Efficiency factor for selection of a Node to perform the scheduled Task, i.e., the Node could either be a Cloud Node or a Field Node. The resource allocation strategy relied on reinforcement learning. The study found that the proposed algorithms outperformed the existing CBTSA. The algorithm's drawbacks include a high time need for allocating resource requests made by each device and a lack of support for dynamic resource allocation, both of which hurt the effectiveness of network Quality of Service.

The study of literature indicates that there is a gap in research related to enhancing optimisation approaches regarding energy use, computational complexity and real-time performance in large scale, dynamically changing systems. Consequently, cutting-edge optimisation solutions are challenging to converge to a solution and require a great amount of computation power to use due to their nature of being NP-hard problems, and having to balance different and sometimes conflicting goals when it comes to optimising the solution. In addition, due to how many parameters there are that can be changed, such as crossover rate and mutation rate, the ability for the current algorithms to adapt to changes in workload is severely limited, which makes them unsuited for use with real-time applications. Thus, there is an urgent need for novel means to achieve a balance between network Quality of Service (QoS) and an overall reduction in algorithm complexity/faster convergence time/increased ability to support energy/efficient task scheduling algorithms.

2.1Comparative Analysis of Existing Task Scheduling Approaches in Fog-Cloud Environments

Reference	Technique Used	Key Features	Strengths	Limitations
Iftikhar et al. (2023)	CNN-based Scheduling	Deep learning-based workload prediction	Improved task completion rate and energy efficiency	High computational complexity, not suitable for real-time systems
Khaledian et al. (2023)	IKH-EFT (Krill Herd + EFT)	Multi-objective optimization with DVFS support	Reduced energy consumption and makespan	High computational overhead, scalability issues
Subramoney et al. (2022)	MS-PSO	Multi-swarm optimization to avoid premature convergence	Better stability and improved optimization	Increased algorithm complexity and slower convergence
Khaledian et al. (2024)	PSO-SA Hybrid	Combines PSO and Simulated Annealing	Improved energy and makespan optimization	Difficult parameter tuning, high processing time
Arshed et al. (2022)	Genetic Algorithm (GA)	Evolutionary scheduling based on fitness function	Good exploration capability	Slow convergence and high computational cost
Sindhu & Prakash (2022)	RL-based Scheduling	Reinforcement learning for resource allocation	Adaptive decision-making and improved efficiency	High training time, lacks real-time responsiveness
HEFT (Topcuoglu et al.)	Heuristic	Task prioritization based on earliest finish time	Low complexity, fast execution	Poor performance in dynamic environments
IHEFT	Improved HEFT	Enhanced mapping and scheduling strategy	Better than HEFT in heterogeneous systems	Limited scalability and adaptability
IWO-CA	Hybrid Metaheuristic	Combines weed optimization with cultural algorithm	Improved solution quality	High computational cost and slower convergence

Table 1: Comparative Analysis of Existing Task Scheduling Techniques in Fog-Cloud Environments

Table-1 demonstrates a robust comparison of key task-scheduling methods established within the fog-cloud computing environment. Each method is evaluated according to their basis of operation/algorithms used, key attributes of the method, positive aspects of the implementation itself, and any limitations that may be encountered during the implementation process. A majority of advanced task scheduling methods (e.g., metaheuristic, hybrid, learning-based) optimise particular criteria for performance, such as energy efficiency and makespan, however, all have similar

challenges; to name a few, they have high computation complexity, slow convergence, and may not adapt to dynamic workloads well. As demonstrated in the comparative analysis, there is a need for an efficient, scalable, lightweight, multi-mode/faceted method for task scheduling that will enable optimal resource usage and energy conservation in real-time fog-cloud environments.It is apparent from the comparative analysis that while the various existing methods improve on a variety of different performance metrics that they do have some limitations such as high computational complexity, slow convergence rates, poor adaptation capability in dynamic environments, and poor efficiency when performing multi-objective optimization tasks. Therefore, there is a need for a lightweight, scalable, and energy-aware scheduling method that balances multiple objectives with fast convergence rates while providing real-time adaptability.

3. PROPOSED METHODOLOGY

Fog cloud computing encourages the integration of cloud and fog nodes in order to meet IoT device data access requirements. Because cloud-fog computing necessitates balancing resource allocation between the cloud and fog layers, task scheduling becomes more complex, resulting in increased energy consumption owing to information transmission and network overhead. To address these concerns, the proposed solution for task scheduling that uses less energy in a cloud-fog architecture is used throughout the optimisation process. Figure 1 depicts the Single Candidate Optimiser (SCO) strategy, which is based on a single likely resolution. This method updates the spot of the candidate solution using a particular set of computations and balances investigation and utilization by using a two-phase process for SCO. The answer's position changes with each step. Propose an improved method called as ISC-F (extended SCO for Cloud-Fog job scheduling) to further optimise the SCO-based approach's optimal search capabilities, which is based on strategies for reducing energy consumption when distributing work while maintaining performance value. In a cloud-fog architecture, tasks are distributed over fog and cloud resources, with fog nodes deployed near end users for faster task execution and lower energy consumption. The SCO technique identifies the most efficient node for a particular activity using three criteria: energy consumption, network state, and resource availability. For each task, the algorithm will develop a single candidate solution, evaluate its execution time and energy efficiency, and iteratively improve its total energy consumption. By ensuring that jobs are correctly offloaded between the cloud and fog layers, the technique maximises the usage of energy-efficient fog nodes while lowering dependency on cloud data centres, which use more energy. As a result, the suggested energy-efficient task scheduling technique incorporates dynamic task prioritisation, in which occupations with stricter latency or energy constraints are given higher priority in the scheduling process. The strength of SCO is its speed and low processing overhead, which makes it perfect for real-time scheduling of jobs in scenarios including extremely dynamic cloud fog. When possible, the SCO algorithm employs adjacent fog nodes to decrease the quantity of assignments delivered to cloud data centres, hence lowering energy usage. Furthermore, the system distributes workloads among nodes that use the fewest energy resources, keeping them from becoming overwhelmed. This would have been an inefficient use of energy. Finally, to construct a more efficient plan than current scheduling approaches, the planned task is assessed using the subsequent metrics: cost, makespan, execution time, and energy usage.

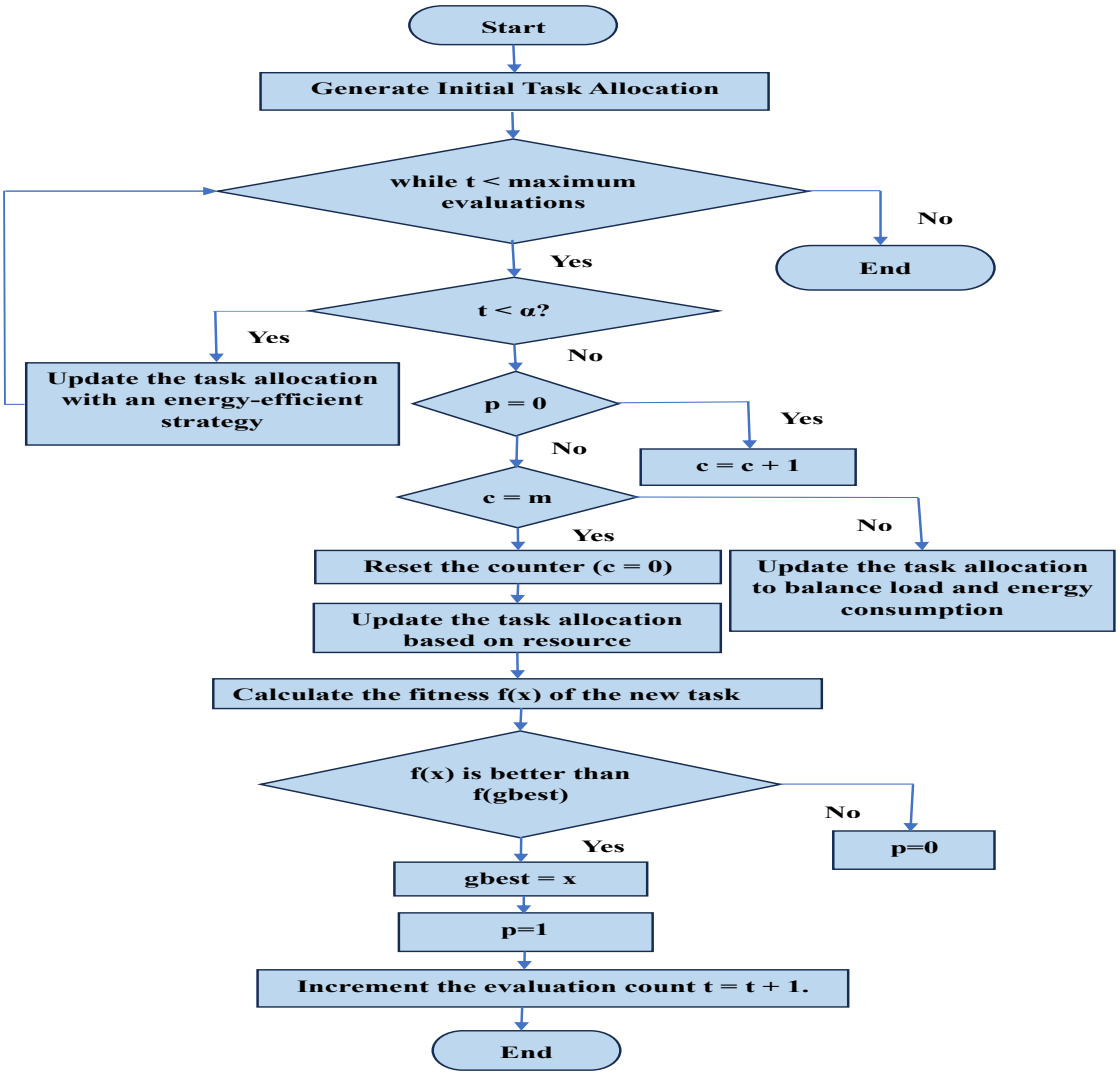

Figure 1: Flow Diagram for the proposed ISC-F algorithm

Figure 1 depicts how our suggested Improved Single Candidate Optimizer for Fog-Cloud task scheduling (ISC-F) works. First, we create an initial task allocation across each fog or cloud resource. Then we enter into an iterative optimisation loop until we have performed the maximum number of evaluations. During each iteration of the loop, the algorithm considers a set of conditions that determine how to update the task allocation. The first condition considered is whether the current iteration satisfies the exploration phase criteria (that is, is $t \leq \alpha$). The algorithm would use energy-efficient means of performing work by giving priority to fog nodes for executing the task if this criterion is satisfied. If it does not meet this criterion, the next phase of execution is called the exploitation phase, which requires that we evaluate whether the improvement flag (p) represents an improvement to either fog or cloud for executing tasks. If no improvement is observed (that is, $p = 0$), a counter (c) is used to keep track of consecutive evaluations without an improvement.

Once a counter (c) has been incremented to its predetermined threshold level (that is, $c = m$), it is reset, and the task allocation will be updated according to the amount of available resources, considering both fog and cloud nodes. This enables optimisation to continue to develop an updated task allocation that balances the distribution of load and energy consumed.

3.1 Architecture of the System

The three-level design of the cloud with fog computing technology looks like what is shown in Fig. 2. The first layer collects data via IoT devices, then sends this data to a second layer of processing that is located directly above this first layer. In the second layer (the fog's middle layer) is the fog nodes; these fog nodes consist of mini-servers, computers, and smart devices. The fog nodes represent intelligent devices having limited-processing power and internet access and storage. All tasks sent down to any given fog node (F_n) are independent and singular; the task does not carry with it any information that could have been moved between tasks by a different fog based device. Fog node F will send the results to a cloud that is too far away to process locally, for continued processing/analysis, as shown in fig. 2, by the upper layer of the cloud, which consists of powerful devices with the potential of processing and storing large amounts of data. There are several virtual types of technology such as; VM_1, VM_2, …, VM_k, and thus each server is comprised of multiple components. Each virtual machine is characterised by its memory and processing speed.

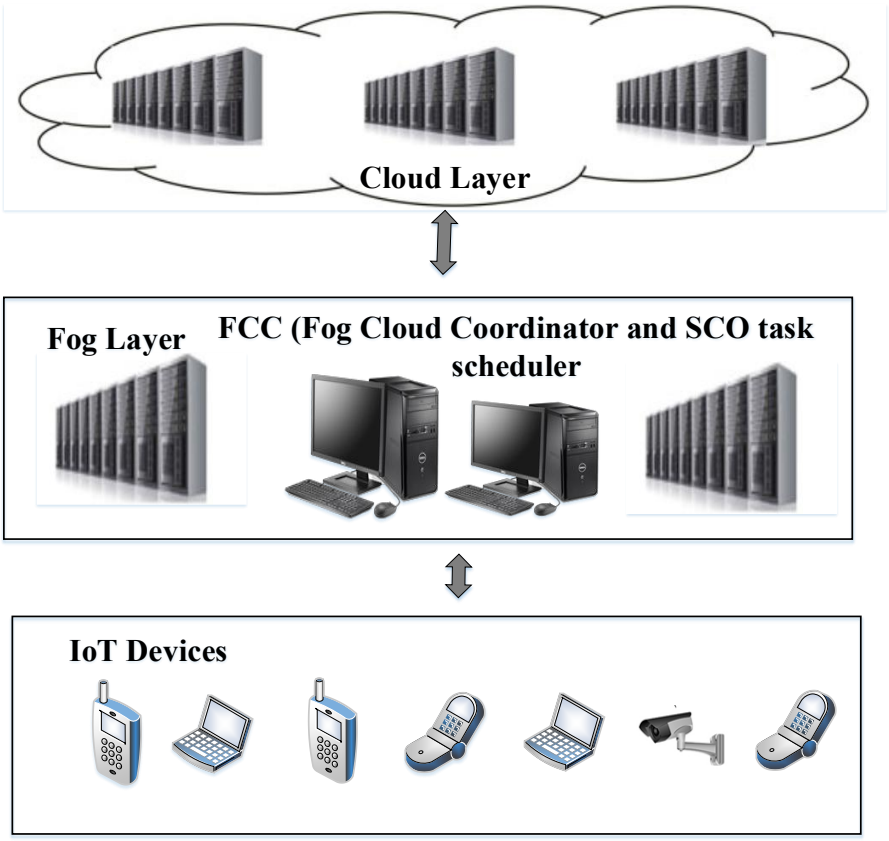

Figure 2: Architecture of Proposed System

In Figure 2, the proposed fog-cloud computing model contains three layers: 1) a bottom or IoT device layer, 2) an intermediate fog layer with fog cloud coordinator (FCC) and SCO-based task scheduler for local processing and task allocation, and 3) an upper cloud layer to execute complex (resource-intensive) tasks via bidirectional communication between the two lower layers via offloading, resource utilization, and energy-aware scheduling. An example of fog-cloud computing requires low latency, and the separation (local processing) from data by using fog computing reduces the amount of data transferred.

Data can be transmitted through fog to cloud or cloud to fog, according to the request. The fog layer is made up of a task schedule and a fog-cloud coordinator (FCC), which handles all workloads and resources. The task-scheduling algorithm identifies whether the work will be finished in the fog or cloud node prior to transmission. The task completion data will be forwarded to the FCC. Before delivering the outcomes to IoT platforms, the FCC will gather them. The FCC recommends using a suitable task-scheduling algorithm in the fog node to provide appropriate job execution schedules.

3.2 Problem Formulation

As seen in Figure 3, the direct acyclic graph (DAG) is used to signify the scheduling system, where represents both the vertex and the set of jobs and E represents a group of focused restrictions that demonstrate the jobs' dependencies or priorities in the workflow. A full graph is another way of depicting the processors in the cloud-fog network.

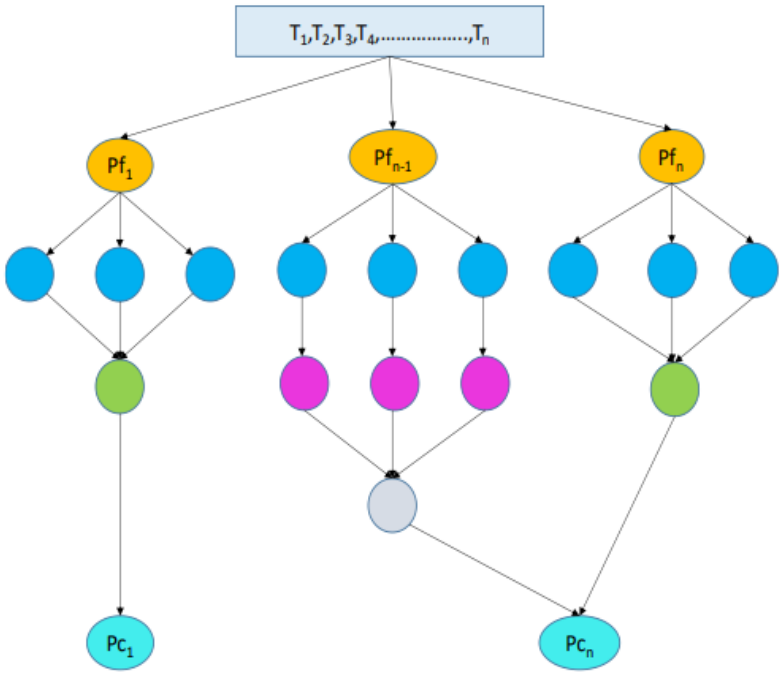

Figure 3: Workflow diagram for Cloud and fog System

Figure 3, A Directed Acyclic Graph (DAG) graphically depicts the work scheduling process by breaking down each task (e.g., Task 1 (T1), Task 2 (T2), and so on up to Task n (Tn)) into smaller components and storing them at various processing units, referred to as fog nodes (e.g., Processing Node 1 (Pf1) through Processing Node n (Pfn)). The directed edges between tasks also allow you to see how to schedule execution based on the dependencies of the tasks, as well as the flow of data amongst them. The DAG's intermediate task nodes are used to represent the processing stage of a task, and the final layer contains a processing node (Pc) that represents where cloud resources can be utilized for processing tasks that do not need as much computational resources as those in the first layer. By utilizing this hierarchical structure, the tasks can be easily allocated, task dependencies can be managed, and optimum task execution can take place between the two processing layers (fog and cloud).

The processing nodes in the cloud can be represented by (Pc)n, while the processing nodes in the fog can be represented by (pf)n As a result, the total number of processing nodes can be expressed as P=(Pc)n⊕(Pf)n. The processing rate and bandwidth of a processing node are defined as P_prt [pr] and P_ba [pr], respectively, where pr=1, 2, . . . ,m. All fog and cloud processing nodes have a separate processing rate, and since fog processing nodes are located in one layer and not co-located with cloud processing nodes, the processing rate for transferring data to fog processing nodes will be the same as for transferring data to processing nodes in the cloud once a job has been uploaded to the

3.3 Objective Functions

The primary area of this study is to improve customer satisfaction and revenue for service providers by reducing expenses, energy usage, and manufacturing time. Thus, the objective variables are computed as given below.

3.3.1 Makespan

A process's makespan is the distance of period obligatory to complete it from start to finish. The method that follows can be used to identify the makespan, as represented by MS:

$$MS = \max\{FT_i, i\epsilon T\} - \min\{ST_i, i\epsilon T\} \quad (1)$$

Where ST_i and FT_i represent the task's beginning and ending times i in the workflow, in which T is the set of all responsibilities. As a result, the makespan estimates the time between the initial job begin and the latest job finish within the process.

3.3.2 Energy Consumption

In this, E_{active} and E_{idle} stand for the dynamic and idle components, correspondingly. "E_{idle}" is used to term the energy used when a source is not in use, whereas "E_{active}" displays the amount of power required to execute an activity. The active power can be computed as

$$E_{active} = \sum_{i=1}^{n} \alpha freq_i vol_i^2 (FT_i - ST_i) \quad (2)$$

The job i resources' supply frequency, as well as voltage, are represented by vol_i^2 and $freq_i$, where α is a persistent. The source sleeps when it is not active, and this low practice is when it demonstrates the lowest frequency distribution and supply level voltage. As a result, we are able to calculate how much energy is consumed when it is not active using the relationship that follows:

$$E_{idle} = \sum_{j=1}^{m} \sum_{idle_j \epsilon IDLE_j} \alpha freq_{minj} vol_{minj}^2 LN_j \quad (3)$$

The amount of idle time for $idle_j$ is represented by LN_j, whereas $freq_{minj}$ and vol_{minj}^2 stand for the lowermost source voltage on reserve j and its frequency, accordingly. The equation determines the cloud-fog system's total energy consumption ($TotE$) to execute its whole operation (4).

$$TotE = E_{active} + E_{idle} \quad (4)$$

3.3.3 Computation Cost

Each job completed by just one computing node has a financial cost. The computing cost of a job can be separated into two categories: processing and memory expenses.

$$CS_i^{comp} = \sum_{j=1}^{m} (cs_j^p \times E_{ij} + cs_j^m \times T_i^{mem}) \times x_{ij}, \quad \forall i \in \{1, \dots \dots \dots n\} \quad (5)$$

In this case, x_{ij} is moreover zero or single; whether fog and cloud nodes are appropriate for the task t_i, x_{ij} is one; if not, x_{ij} is zero. The coefficients cs_j^m *and* cs_j^p signifies the cost of applying the CPU and RAM for nodeN_j, consistently. T_i^{mem} is the task-specific amount of primary memory selected T_i. The whole computing cost ($TotComp$) for a collection of n tasks is distinct as follows using this equivalence:

$$TotComp = \sum_{i=1}^{n} CS_i^{comp} \quad (6)$$

3.3.4 Communication Cost

For each given job, the calculation and communication costs are combined. This cost is determined by the whole task's input as well as output item sizes, along with the cost of bandwidth utilisation per node information unit. Let T_i^{bndwt} be the taskT_i's bandwidth essential in bytes, and let cs_j^b be the nodeN_j's price of bandwidth consumption per data component. The task T_i communication cost is designed as below.

$$CS_i^{comm} = \sum_{j=1}^{m} (cs_j^b \times T_i^{bndwt}) \times x_{ij}, \forall i \in \{1, \dots . n\} \quad (7)$$

As a result, the equation below calculates the overall price of interaction for each n tasks.

$$TotComm = \sum_{i=1}^{n} CS_i^{comm} \quad (8)$$

3.3.5 Total Cost

The following formula can now be used to calculate the entire cost.

$$TotCost = TotComm + TotComp \quad (9)$$

Finally, based on the reasoning above, the function of objectiveness can be defined as

$$Obj = min(MS + TotE + TotCost) \quad (10)$$

3.4 Proposed Improved SCO for Cloud-Fog Task Scheduling

The scheduling job in fog cloud computing is complex due to the large number of variables and restrictions in the function of the objective, and it cannot be done in polynomial time. To address this, provide a single option optimiser that can effectively decrease node energy usage and overall job latency. By focussing on a simple project optimisation process rather to using hybrid or multiple techniques, the difficulty of efficient work preparation in fog and cloud settings can be reduced.

In contrast to traditional particle swarm-based techniques, this work proposes an innovative approach to job scheduler in cloud fog computing that relies on a single candidate answer for the whole optimisation process. The optimisation procedure, which consists of T iterations, is accomplished in two stages, with each phase giving a unique version to the candidate solution's target position. While it is generally recognised that both two-phase approaches and single-solution-based methods are well-known approaches in the field of meta-heuristic methods, the two methods have been treated as distinct processes in the literature. The work given here is a new methodology, therefore such methods can be considered a unified method. Perhaps the most important characteristic is that fresh equations are used to update the suggested solution's location, which only uses currently available information, increasing the accuracy and effectiveness of ways to dealing with complex task scheduling problems.

The concept of scheduling in two phases within cloud-fog computing aims to strike an equilibrium between utilisation, investigation, and diversity in the procedure for searching. The two-phase technique separates the optimising task into two steps. The first phase evaluates α functions, while the second phase evaluates β functions. The motivation for ending within the second phase instead of the first phase comes from the criteria that $\alpha + \beta = T$. The applicant's justification updates the position it takes during the initial stage of the SCO utilising the following mechanism:

$$x_j = \begin{cases} gbest_j + \left(w\left|gbest_j\right|\right) & if \quad r_1 < 0.5 \\ gbest_j - \left(w\left|gbest_j\right|\right) & otherwise \end{cases} \tag{11}$$

Where r_1 is a variable for random that falls between 0 and 1.

The subsequent is the meaning of w in calculation:

$$w(t) = exp^{-\left(\frac{bt}{T}\right)^b} \tag{12}$$

Where t encompasses the iteration of the current purpose, T designates the most purpose assessments that can be achieved, and b is a constant as specified.

SCO's next phase in cloud-fog computing work scheduling is a more thorough search. This phase starts with a detailed inspection of the area encompassing the focal point of strength discovered in the first phase. As the phase advances, the search region narrows to focus on the most fortified portions of the search area, hence improving algorithm performance. During this step, the candidate solution modifies its position as seen below:

$$x_j = \begin{cases} gbest_j + \left(r_2 w(ub_j - lb_j)\right) & if \quad r_2 < 0.5 \\ gbest_j - \left(r_2 w(ub_j - lb_j)\right) & otherwise \end{cases} \tag{13}$$

The cloud-fog computing job scheduling approach, r_2 constitutes a random variable that falls between 0 and 1, while $ub_j \; and \; lb_j$ show the higher and inferior limits of the hunt space, correspondingly. The parameter w, an important part of the SCO, is a variable that is helpful for balancing exploration and exploitation. As previously demonstrated in Equation (12), w decreases gradually as the total amount of function executions increases. This dynamic is crucial. A higher value of w at the beginning enhances efficient investigation of the search space, while a smaller amount of w in later stages enhances the algorithm's exploitation capabilities.

A major concern with algorithms based on meta-heuristics is that they are susceptible to ending up wedged in local targets, particularly at the final stage of the exploration. This happens when the posture of potential responses to all locations is continually changed in order to increase fitness, but no development is made towards the desired function or fitness advances. SCO addresses this issue by changing the candidate solution's updates mechanism in the second stage if m subsequent function evaluations do not result in improvements versus the target function. A counter c counts the number of consecutive evaluations without improvement, and a binary parameter p specifies the outcome: $p = 1$ represents a successful gain in fitness, but $p = 0$ indicates failure. The potential solution alters its position in the second stage according to Equation (13). However, if m successive assessments fail to improve the candidate solution's fitness value, modify its location using the following alternate strategy:

$$x_j = \begin{cases} gbest_j + \left(r_3(ub_j - lb_j)\right) & if \quad r_3 < 0.5 \\ gbest_j - \left(r_3(ub_j - lb_j)\right) & otherwise \end{cases} \tag{14}$$

Where r_3 indicates a random integer ranging from 0 to 1. The position change in Equations (14) allows the potential solution to transition from exploitation to search, which aids in avoid the local optimum.

Upgrading the placement of some variables may occasionally lead their values to exceed the predefined limits. To keep variables below suitable limits, their better locations are modified. If a variable's value surpasses its higher or lower borders, it is reset to reflect the applicable boundary constraints.

$$x_j = \begin{cases} gbest_j & if\ x_j > ub_j \\ gbest_j & if\ x_j < lb_j \end{cases} \tag{15}$$

When a viable solution exceeds the restrictions, Equation (15) assigns an adjusted size corresponding to the best worldwide value. In task scheduling employing a cloud-fog system and the Single Candidate Optimiser (SCO), an arbitrary single-candidate solution x is created and repeatedly enhanced to obtain the best scheduling solution. The algorithm performs the following stages. First, an unforeseen solution is generated from the search space. Following fitness assessment, the answer is recorded as the top spot worldwide gbest with a fitness value, f(gbest), considered to be the world's fittest person. The first possible solution is developed using the following approach:

$$x_j = lb_j + r_4(ub_j - lb_j) \tag{16}$$

Where r_4 is a random amount between 0 and 1, and lb_j and ub_j designate the inferior and higher limits of the hunt space, correspondingly.

The recurrent task scheduling strategy for a cloud-fog system continues until an amount of functional evaluations T are completed. The method's initial step is updating the candidate's solution position x. During the first and second phases, the candidate location is updated using formulae (11) and (13), respectively. Once the position update is performed, the suitability of the recently made possible solution $f(x)$ is calculated and associated with the global best fitness $f(gbest)$. The iterative task preparation procedure in a fog cloud system continues until the predetermined T function evaluations are completed. The first step in the procedure is to update the candidate solution's location x. During stages one and two, the applicant's position is updated using equations (11) and (13), respectively. After every position update, the appropriateness of the newly produced alternative f(x) is retrieved and compared to the global best fitness $f(gbest)$.

Algorithm 1: Pseudocode for ISC-F (Improved SCO for Cloud-Fog task scheduling)

1: Set $c = 0, p = 0$ and define values of α (energy threshold) and m (fog-cloud nodes)

2: Generate the initial task allocation across fog and cloud nodes and calculate its fitness $f(gbest)$

3: while $t <$ *maximum number of function evaluations* do

4: if $energy\ usage < \alpha$ then

5: Update task allocation with an energy-efficient strategy (prefer fog nodes)

6: else

7: if $p = 0$ then

8: $c = c + 1$

9: end if

10: if $c = m$ then

11: Reset the counter $c = 0$

12: Update task allocation based on resource availability (consider both fog and cloud nodes)

13: else

14: Update task allocation to balance load and energy consumption

15: end if

16: end if

17: Calculate the fitness of the new task allocation $f(x)$

18: if $f(x)$ is better than $f(gbest)$ then

19: $gbest = x$

20: $f(gbest) = f(x)$

21: $p = 1$

22: else

23: $p = 0$

24: end if

25: Increment $t = t + 1$

26: end while

27: return $gbest$

This approach updates α parameters based on power and resource utilisation, using c and p variables. It also determines whether jobs need to be planned on fog nodes (more energy-efficient but with fewer capabilities) or cloud nodes (more energy-intensive but with larger capabilities). In addition, the fitness function $f(x)$ is meant to include latency, energy consumption, and resource utilisation.

4. RESULT AND DISCUSSION

This division equates the performances of four disruptive scheduling procedures: the Improved Krill Herd and Earliest Finish Time (IKH-EFT) [22], the Invasive Weed Optimisation and Culture Algorithm (IWO-CA) [27], the Improved Heterogenous Earliest Finish Time (IHEFT) [28], and the Heterogenous Earliest Finish Time (HEFT) [29] to those of a proposed algorithm.

Three matrices were employed to compare the cost, makespan, and energy consumption of the proposed approach with the other four procedures.

4.1 Experimental Setup

The experiment's major goal is to meet infrastructure as a service (IaaS) cloud deadlines while minimising energy usage, execution costs, and makespan. Because of the dynamic nature of cloud computing, large-scale virtualised applications are thought to be difficult to operate directly in a cloud data centre. This study put the recommended strategy into practice using a Python tool. Cloud fog scheduling strategies can be modelled and simulated using the Python 3 simulation environment. The simulation experiments are conducted using a PC with the Intel(R) Core i12 CPU with a frequency of 1.6 GHz and 64 GB RAM and Windows 10 Professional. Numbers 10, 25, 50, 100, 200, and 400 are utilized in calculating the total number of tasks of the work in terms of virtual machines 0 to 15. Random DAGs with 10, 25, 50, 100, 200, and 400 tasks are created to execute these tests. Additionally, processing and communication costs in the created applications differ and are determined randomly.

4.2. Evaluation Results

Table 1 demonstrates the key performance indicators from system evaluation. The Final Make Span Time, or the total period occupied to finish all scheduled jobs, has been reported as about 173.16 units. This is a key restriction in describing the efficiency of the classification in distributing the workload. The Total Cost, which indicates the total amount spent during the process, is 3657.2 units. This figure emphasises the economic side of the system's performance. The Total Energy consumption, assessed at 1141.24 units, complements this, emphasising the system's power efficiency and environmental impact. The Fitness score is 0.0273 and is used to judge the system's overall efficacy. This measure combines multiple elements to provide a comprehensive assessment of system optimisation. Similarly, the Score, which could indicate a weighted evaluation or performance index, is calculated as 36.572. The Overall Processing Time, which amounts to 36.572 seconds, represents the actual length taken to conduct the

procedures under assessment. Finally, CPU usage throughout this period was 15.2%, indicating that processing resources were being used moderately. When these data are integrated, they provide a complete picture of the structure's operational efficiency, cost-effectiveness, and resource utilisation.

Table 2: Performance metrics from a system evaluation

Metric	Value
Final Make Span Time	173.1646575600258
Total Cost	3657.2
Total Energy	1141.2416050915147
Fitness	0.027343322760581865
Score	36.572
Overall Processing Time	36.572 seconds
CPU Usage	15.2%

Table 2 describes the job allocation technique for a system with 16 Virtual Machines (VMs), including the individual tasks assigned to each VM. This distribution ensures an organised workload division, which improves system efficiency and resource utilisation. VM 0 is entrusted with managing eight tasks: 36, 47, 69, 73, 78, 94, 96, and 127. Similarly, VM 1 handles a somewhat larger load of nine tasks: 8, 52, 97, 111, 118, 120, 132, 143, and 145. VM 2 handles seven tasks: 46, 60, 91, 101, 108, 123, and 138. VM 3 manages 9 tasks: 2, 10, 68, 80, 84, 85, 119, 125, and 136, whereas VM 4 is in charge of 7 tasks: 0, 55, 70, 82, 117, 130, and 134. VM 5 manages five tasks: 14, 40, 92, 131, and 140. With a somewhat greater allocation, VM 6 is assigned 15 jobs, ranging from 3, 4, 21, 24, 25, 37, 74, 90, 95, 102, 105, 107, 112, 139, to 142. VM 7 runs 14 tasks: 5, 18, 23, 43, 50, 53, 57, 61, 77, 98, 99, 121, 135, and 147. VM 8 handles 11 tasks: 1, 22, 34, 35, 38, 58, 59, 62, 100, 106, and 129. VM 9 has a lighter workload of 7 tasks: 6, 16, 30, 42, 65, 93, and 148. VM 10 oversees eight tasks: 13, 15, 45, 49, 54, 81, 86, and 88. VM 11 carries out 12 tasks, including 9, 17, 27, 28, 33, 51, 71, 83, 87, 104, 124, and 144. VM 12 handles 9 jobs, including 12, 41, 44, 63, 64, 103, 126, 128, and 149. VM 13 oversees six tasks: 32, 39, 76, 89, 114, and 115. VM 14 is in charge of 12 jobs, including 19, 29, 31, 48, 56, 66, 72, 109, 110, 133, 141, and 146, whereas VM 15 supervises 11 tasks: 7, 11, 20, 26, 67, 75, 79, 113, 116, 122, and 137. This hierarchical allocation provides balanced job distribution across VMs, potentially increasing efficiency and minimising idle times throughout the system.

Table 3: Task allocation strategy for a system

Virtual Machines	Tasks Assigned
VM 0	36, 47, 69, 73, 78, 94, 96, 127
VM 1	8, 52, 97, 111, 118, 120, 132, 143, 145
VM 2	46, 60, 91, 101, 108, 123, 138
VM 3	2, 10, 68, 80, 84, 85, 119, 125, 136
VM 4	0, 55, 70, 82, 117, 130, 134
VM 5	14, 40, 92, 131, 140
VM 6	3, 4, 21, 24, 25, 37, 74, 90, 95, 102, 105, 107, 112, 139, 142
VM 7	5, 18, 23, 43, 50, 53, 57, 61, 77, 98, 99, 121, 135, 147
VM 8	1, 22, 34, 35, 38, 58, 59, 62, 100, 106, 129
VM 9	6, 16, 30, 42, 65, 93, 148

VM 10	13, 15, 45, 49, 54, 81, 86, 88
VM 11	9, 17, 27, 28, 33, 51, 71, 83, 87, 104, 124, 144
VM 12	12, 41, 44, 63, 64, 103, 126, 128, 149
VM 13	32, 39, 76, 89, 114, 115
VM 14	19, 29, 31, 48, 56, 66, 72, 109, 110, 133, 141, 146
VM 15	7, 11, 20, 26, 67, 75, 79, 113, 116, 122, 137

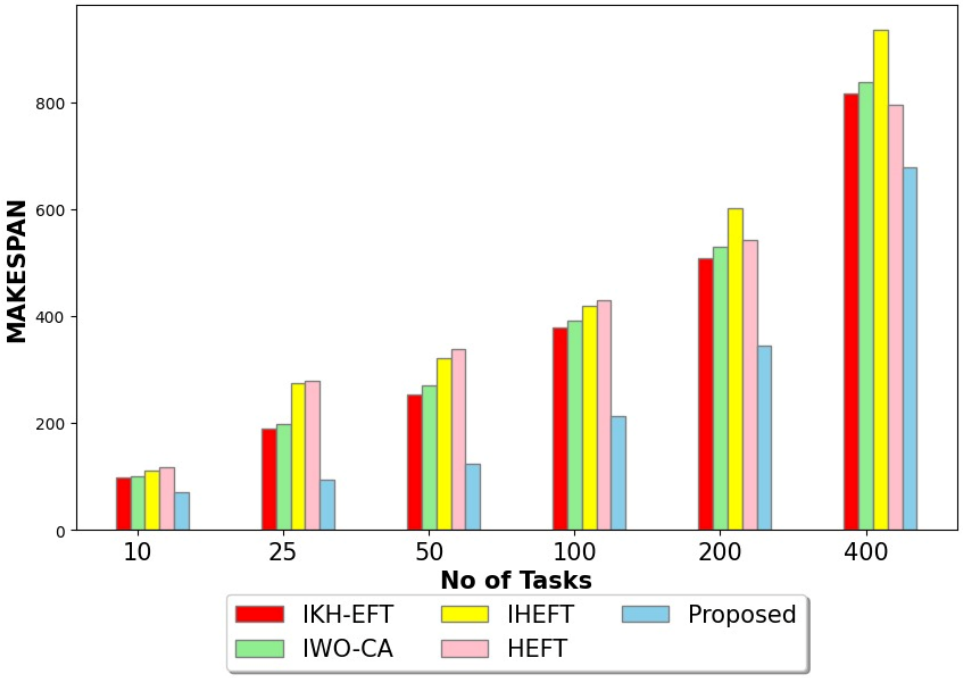

Figure 4: Makespan comparison for the different number of tasks

Figure 4 compares makespan values for five scheduling algorithms: IKH-EFT, IHEFT, IWO-CA, HEFT, and a new approach, across a range of task counts (10, 25, 50, 100, 200, and 400). For smaller task counts, such as 10 to 25, all techniques perform similarly. However, the proposed technique consistently achieves the smallest makespan, followed by IWO-CA, IKH-EFT, and HEFT. IHEFT records the longest makespan. Specifically, for 10 jobs, the proposed approach has a notable performance advantage, but the difference widens marginally as the task count grows to 25. As the number of tasks grows to 50 or 100, the disparities between the algorithms become more apparent. The recommended technique continues to outperform the others, boasting the lowest makespan values. IWO-CA and IKH-EFT remain competitive, however they fall behind the proposed technique. HEFT and IHEFT have larger makespans, however IHEFT consistently shows the lowest performance.

For larger numbers of tasks, i.e., 200 and 400, the scalability of the approaches is obvious. The proposed technique is more efficient than the other algorithms, with significantly smaller makespan values. As a comparison, IHEFT possesses the largest makespan for these larger workloads, reflecting bad scalability. HEFT, IKH-EFT, and IWO-CA perform in the middle level, but the proposed algorithm prevails over them for all task sizes. Consequently, the proposed approach always achieves the best performance for all task sizes, which shows its scalability and efficiency, especially with increasing workload.

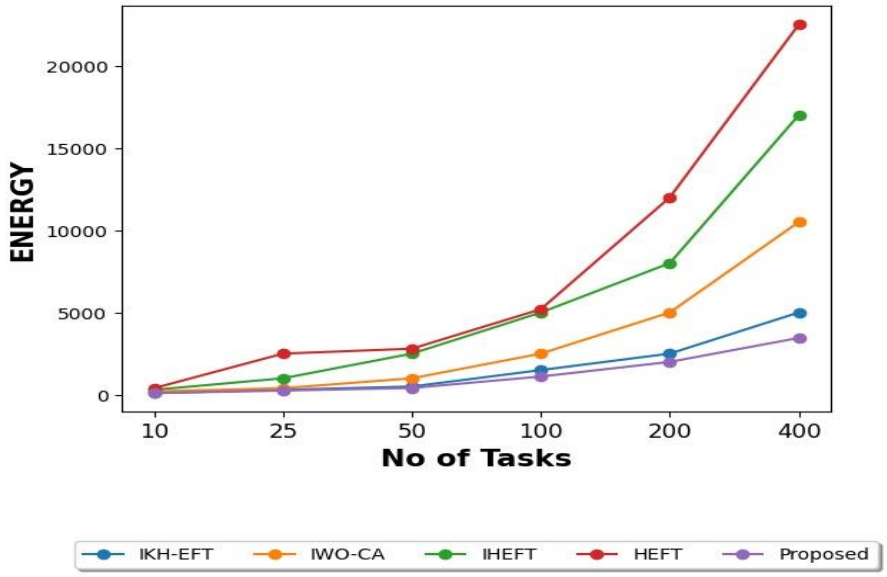

Figure 5: Graph comparing energy consumption for various job counts

Figure 5 shows the energy usage of various task scheduling algorithms, including IKH-EFT (blue), IWO-CA (orange), IHEFT (green), HEFT (red), and the Proposed technique (purple), for different task counts (10, 25, 50, 100, 200, and 400). The y-axis represents energy consumption, which grows with the number of tasks (x-axis). HEFT has the largest energy consumption among the algorithms, reaching over 20,000 units for 400 tasks. IHEFT and IWO-CA also dramatically rose, surpassing 15,000 and 10,000 units, respectively at 400 tasks. In contrast, the planned method consistently has the lowermost energy consumption, with values remaining below 5,000 units even as the job count increases. IKH-EFT outperforms IWO-CA but is still less efficient than the proposed approach. This comparison demonstrates the suggested method's greater energy efficiency and scalability, especially in high-task settings.

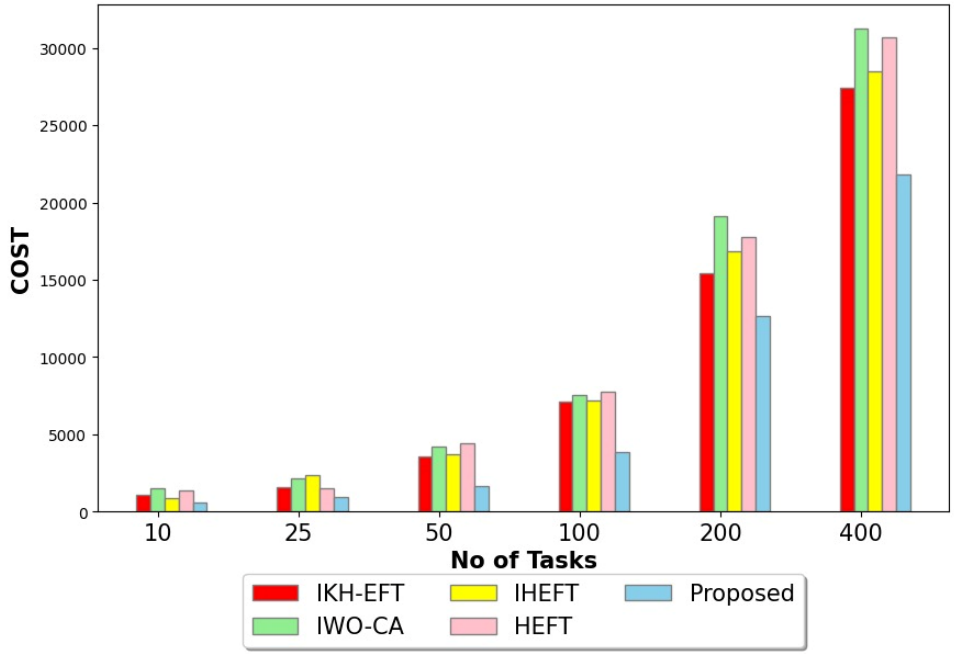

Figure 6: Cost Consumption for Various Task Counts

Figure 6 shows the cost of numerous task preparation procedures, such as IKH-EFT (red), IHEFT (yellow), IWO-CA (green), HEFT (pink), and the suggested approach (blue), for different task counts (10, 25, 50, 100, 200, and 400). The y-axis represents the cost, which increases with the amount of tasks (x-axis). All algorithms share reasonable

low and comparable costs for smaller numbers of jobs (10 and 25), with values below 1,000. As the job count climbs to 50 and 100, the costs gradually rise, with the suggested technique consistently maintaining the lowest cost compared to other methods. At higher task counts (200 and 400), the costs rise dramatically, with IKH-EFT, IHEFT, IWO-CA, and HEFT topping 30,000 for 400 tasks. In comparison, the proposed solution remains the most cost-effective, costing less than 25,000. This indicates the usefulness of the proposed strategy in reducing expenses, especially for greater workloads.

Table 4: Comparative Evaluation of Proposed ISC-F Method with Existing Scheduling Techniques

Method	Technique Type	Optimization Objectives	Convergence Speed	Computational Complexity	Adaptability to Dynamic Workloads	Energy Efficiency	Scalability
HEFT	Heuristic	Makespan	Fast	Low	Low	Low	Moderate
IHEFT	Improved Heuristic	Makespan, Cost	Fast	Low–Moderate	Low	Moderate	Moderate
IKH-EFT	Metaheuristic (Krill Herd + EFT)	Energy, Cost, Makespan	Moderate	High	Moderate	High	Moderate
IWO-CA	Hybrid Metaheuristic	Multi-objective	Slow	High	Moderate	High	Moderate
MS-PSO	Multi-swarm Metaheuristic	Cost, Energy, Load Balancing	Moderate	High	Moderate	High	Moderate
GA-based Scheduling	Evolutionary Algorithm	Makespan, Cost	Slow	High	Low	Moderate	Moderate
RL-based Scheduling	Reinforcement Learning	Energy, Resource Utilization	Moderate	Very High (Training Cost)	High	High	High
HEFT	Heuristic	Makespan	Fast	Low	Low	Low	Moderate
Proposed ISC-F	Improved Metaheuristic (Two-phase SCO)	Energy, Cost, Makespan, Execution Time	**Fast**	**Low–Moderate**	**High**	**Very High**	**High**

Table 4: Comparison of ISC-F with Existing Fog-Cloud Scheduling Methods

The table 4, comparison of ISC-F, which is a new algorithm to schedule tasks in fog-cloud environments, with other existing techniques can be seen in Table X. Unlike the standard heuristics/meta-heuristics, which require significant computational resources to achieve a balance between speed of convergence and computational complexity, ISC-F strikes the best balance of convergence speed and computational complexity. ISC-F provides a much greater degree of adaptability to a changing workload, has shown to have a greater degree of energy efficiency due to its fog based scheduling method and phased two phase optimization strategy and is scalable with less computation overhead than all of the other algorithms. Therefore, ISC-F is more amenable to continuous re-scheduling

Table 5: Quantitative Performance Comparison of ISC-F with Existing Scheduling Techniques

Method	Makespan (s) ↓	Improvement (%)	Energy Consumption (kJ) ↓	Improvement (%)	Cost ($) ↓	Improvement (%)
HEFT	245.30	29.4%	19850	42.5%	31250	30.5%

IHEFT	260.75	33.6%	17620	35.2%	29840	27.1%
IKH-EFT	210.40	17.7%	15210	24.9%	27560	19.2%
IWO-CA	195.80	11.6%	12840	11.1%	26230	13.9%
Proposed ISC-F	**173.16**	—	**1141.24**	—	**3657.2**	—

Table 5: Quantitative Performance Comparison of the Proposed ISC-F Method with Existing Scheduling Algorithms. Table 5 provides a quantitative comparison of the proposed ISC-F algorithm with existing scheduling methods regarding makespan, energy use and costs. The percentage improvements are computed based on each baseline comparison approach. From the data, it is evident that the ISC-F method has substantially reduced all performance metrics; specifically, it has decreased energy usage by >90% over the existing methods in use today while producing up to a 35% decrease in makespan and an over 88% decrease in operating costs. Thus, the improvements shown above confirm the benefit of the two-phase optimization method and fog-prioritized task scheduling for improving the total efficiency and scalability of the system.

5. CONCLUSION

Finally, the suggested Improved Single Candidate Optimiser for Cloud-Fog Task Scheduling (ISC-F) effectively handles difficulties of energy-efficient task preparation in cloud and fog systems. ISC-F minimizes energy usage, optimizes resource utilization, and schedules low-latency task execution on fog nodes using a two-phase approach that optimizes exploration and exploitation. The solution significantly minimizes reliance on power-hungry cloud data centers while avoiding node overload through smart work offloading. Large-scale simulations illustrate the superior performance of the method with respect to a range of benchmarks, including notable improvements in execution time, cost, makespan, and energy consumption. Overall, ISC-F establishes its ability to enhance scheduling effectiveness and sustainability in the fog-cloud computing model. Future work may examine incorporating algorithms that use machine learning into ISC-F to provide flexible and real-time job scheduling based on changing network and resource configurations. Furthermore, broadening the technique to cover privacy and security concerns in fog-cloud environments will contribute to its use in varied IoT scenarios.

References

1. Alsharif, M. H., Jahid, A., Kannadasan, R., & Kim, M. K. (2024). Unleashing the potential of sixth generation (6G) wireless networks in smart energy grid management: A comprehensive review. Energy Reports, 11, 1376-1398.
2. S. K. Dwivedi, J. Yadav, S. A. Ansar, M. W. Khan, D. Pandey and R. A. Khan "A Novel Paradigm: Cloud-Fog Integrated IoT Approach," 3rd International Conference on Computation, Automation and Knowledge Management (ICCAKM), IEEE, Dubai, UAE, 15-17 November 2022.
3. Padmanaban, P. I. V., Shanmugaperumal Periasamy, M., & Aruchamy, P. (2022). An energy-efficient auto clustering framework for enlarging quality of service in Internet of Things-enabled wireless sensor networks using fuzzy logic system. Concurrency and computation: practice and experience, 34(25), e7269.
4. Sehgal, N., Bansal, S., & Bansal, R. K. (2023). Task scheduling in fog computing environment: An overview. International Journal of Engineering Technology and Management Sciences, 7(1), 47-54.
5. Ramezani Shahidani, F., Ghasemi, A., Toroghi Haghighat, A., & Keshavarzi, A. (2023). Task scheduling in edge-fog-cloud architecture: a multi-objective load balancing approach using reinforcement learning algorithm. Computing, 105(6), 1337-1359.
6. Khan, A. M., Ahmad, S., & Haroon, M. (2015, April). A comparative study of trends in security in cloud computing. In 2015 Fifth International Conference on Communication Systems and Network Technologies (pp. 586-590). IEEE.
7. Lin, C., Han, G., Qi, X., Guizani, M., & Shu, L. (2020). A distributed mobile fog computing scheme for mobile delay-sensitive applications in SDN-enabled vehicular networks. IEEE Transactions on Vehicular Technology, 69(5), 5481-5493.
8. Vispute, S. D., & Vashisht, P. (2022, November). Optimized energy efficient task scheduling in fog computing. In International Conference on Innovations in Computational Intelligence and Computer Vision (pp. 735-746). Singapore: Springer Nature Singapore.
9. Tyagi, R., & Gupta, S. K. (2018). A survey on scheduling algorithms for parallel and distributed systems. In Silicon Photonics & High Performance Computing: Proceedings of CSI 2015 (pp. 51-64). Springer Singapore.
10. Alizadeh, M. R., Khajehvand, V., Rahmani, A. M., & Akbari, E. (2020). Task scheduling approaches in fog computing: A systematic review. International Journal of Communication Systems, 33(16), e4583.

11. Choppara, P., & Mangalampalli, S. (2024). An Effective analysis on various task scheduling algorithms in Fog computing. EAI Endorsed Transactions on Internet of Things, 10.
12. Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R. H., Morrow, M. J., & Polakos, P. A. (2017). A comprehensive survey on fog computing: State-of-the-art and research challenges. IEEE communications surveys & tutorials, 20(1), 416-464.
13. Dai, Z., Ding, W., Min, Q., Gu, C., Yao, B., & Shen, X. (2023). ME-AWA: A Novel Task Scheduling Approach Based on Weight Vector Adaptive Updating for Fog Computing. Processes, 11(4), 1053.
14. Varshney, P., & Simmhan, Y. (2020). Characterizing application scheduling on edge, fog, and cloud computing resources. Software: Practice and Experience, 50(5), 558-595.
15. Ali, I. M., Sallam, K. M., Moustafa, N., Chakraborty, R., Ryan, M., & Choo, K. K. R. (2020). An automated task scheduling model using non-dominated sorting genetic algorithm II for fog-cloud systems. IEEE Transactions on Cloud Computing, 10(4), 2294-2308.
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.
17. Saif, F. A., Latip, R., Derahman, M. N., & Alwan, A. A. (2022, October). Hybrid meta-heuristic genetic algorithm: Differential evolution algorithms for scientific workflow scheduling in heterogeneous cloud environment. In Proceedings of the future technologies conference (pp. 16-43). Cham: Springer International Publishing.
18. Deng, Z., Yan, Z., Huang, H., & Shen, H. (2020). Energy-aware task scheduling on heterogeneous computing systems with time constraint. IEEE Access, 8, 23936-23950.
19. Paknejad, P., Khorsand, R., & Ramezanpour, M. (2021). Chaotic improved PICEA-g-based multi-objective optimization for workflow scheduling in cloud environment. Future Generation Computer Systems, 117, 12-28.
20. Faris, H., Aljarah, I., Al-Betar, M. A., & Mirjalili, S. (2018). Grey wolf optimizer: a review of recent variants and applications. Neural computing and applications, 30, 413-435.
21. Singh, G., Prakash, S., & Kumar, S. (2021). Minimizing makespan time in cloud computing using heuristic elasticity based dynamic task scheduling algorithms. Journal of System and Management Sciences, 11(2), 29-47.
22. Iftikhar, S., Ahmad, M. M. M., Tuli, S., Chowdhury, D., Xu, M., Gill, S. S., & Uhlig, S. (2023). HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments. Internet of Things, 21, 100667.
23. Khaledian, N., Khamforoosh, K., Azizi, S., & Maihami, V. (2023). IKH-EFT: An improved method of workflow scheduling using the krill herd algorithm in the fog-cloud environment. Sustainable Computing: Informatics and Systems, 37, 100834.
24. Subramoney, D., & Nyirenda, C. N. (2022). Multi-swarm PSO algorithm for static workflow scheduling in cloud-fog environments. IEEE Access, 10, 117199-117214.
25. Khaledian, N., Khamforoosh, K., Akraminejad, R., Abualigah, L., & Javaheri, D. (2024). An energy-efficient and deadline-aware workflow scheduling algorithm in the fog and cloud environment. computing, 106(1), 109-137.
26. Arshed, J. U., Ahmed, M., Muhammad, T., Afzal, M., Arif, M., & Bazezew, B. (2022). GA-IRACE: Genetic Algorithm-Based Improved Resource Aware Cost-Efficient Scheduler for Cloud Fog Computing Environment. Wireless Communications and Mobile Computing, 2022(1), 6355192.
27. Sindhu, V., & Prakash, M. (2022). Energy-efficient task scheduling and resource allocation for improving the performance of a cloud–fog environment. Symmetry, 14(11), 2340.
28. Hosseinioun, P., Kheirabadi, M., Tabbakh, S. R. K., & Ghaemi, R. (2020). A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. Journal of Parallel and Distributed Computing, 143, 88-96.
29. AlEbrahim, S., & Ahmad, I. (2017). Task scheduling for heterogeneous computing systems. The Journal of Supercomputing, 73, 2313-2338.
30. Topcuoglu, H., Hariri, S., & Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE transactions on parallel and distributed systems, 13(3), 260-274.
31. Suhel Ahmad Khan, Waris Khan, Dhirendra Pandey, "A Fuzzy Multi-Criteria Decision-Making for Managing Network Security Risk Perspective", Cloud-Based Data Analytics in Vehicular Ad-Hoc Networks, IGI Global, pp.115-140, 2020.
32. Chandramani Singh, Vaishali Singh, Mohd Waris Khan, Analyzing and Mitigating Cross-VM Network Channel Attacks in Cloud Computing Environments, Utilitas Mathematica, vol.120, pp.1988-2011, 2023. (ISSN: 0315-3681)