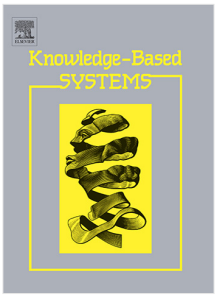
Journal Pre-proof



Multiprocessor task scheduling using multi-objective hybrid genetic Algorithm in Fog-cloud computing

Sachi Gupta, Rakesh Ahuja, Gaurav Agarwal, Atul Kumar Rai

PII: S0950-7051(23)00313-1
DOI: https://doi.org/10.1016/j.knosys.2023.110563
Reference: KNOSYS 110563

To appear in: *Knowledge-Based Systems*

Received date : 25 March 2022
Revised date : 25 March 2023
Accepted date : 10 April 2023

Please cite this article as: S. Gupta, R. Ahuja, G. Agarwal et al., Multiprocessor task scheduling using multi-objective hybrid genetic Algorithm in Fog-cloud computing, *Knowledge-Based Systems* (2023), doi: https://doi.org/10.1016/j.knosys.2023.110563.

Multiprocessor task scheduling using multi-objective hybrid genetic Algorithm in Fog-cloud computing

[*1]Dr. Sachi Gupta, [2]Dr. Rakesh Ahuja, [3]Gaurav Agarwal, [4]Dr. Atul Kumar Rai,

[*1]Department of Information technology,
IMS Engineering College, Ghaziabad, Uttar Pradesh, India.
[2]Chitkara University Institute of Engineering and Technology,
Chitkara University, Punjab, India,
[3]Department of Computer Science & Engineering,
IMS Engineering College, Ghaziabad, Uttar Pradesh, India.
[4]Department of Computer Science & Engineering,
Kothiwal Institute of Technology and Professional Studies, Moradabad, India.
[*]Email: guptasachi356@gmail.com

Abstract: Multiprocessor task scheduling is an operation of processing more than two tasks simultaneously in the system. The Fog-cloud multiprocessor computing structures are the categories of exchanged collateral structures with great demand from its initiation. Like other networking systems, the existing fog cloud system based on multiprocessor systems faces some challenges. Due to the availability of excess clients and various services, scheduling and energy consumption issues are challenging. The existing problems must be resolved with proper planning to reduce makespan and energy consumption. To obtain this, an optimal scheduling approach is required. The proposed approach presents a novel methodology called Hybrid Genetic Algorithm and Energy Conscious Scheduling for better scheduling tasks over the processors. Here Genetic Algorithm and Energy conscious scheduling model are integrated. When only a Genetic Algorithm is chosen for the task scheduling approach, it becomes computationally expensive. Energy consumption becomes a huge challenge as it does not cope with complexity, making it extremely difficult to schedule appropriate tasks. When choosing the proposed hybrid Genetic algorithm, these issues can be overcome by considering optimal solutions with minimized makespan and consumed energy. A Genetic Algorithm is used to generate three primary chromosomes using priority approaches. The allocated resources are optimized through the Energy Conscious Scheduling model, and the proposed method is implemented using MATLAB. The existing methods, including genetic algorithm, particle swarm optimization, gravitational search algorithm, ant colony optimization and round robin models, are compared with the proposed method, proven comparatively better than existing models.

Index terms: Fog-cloud system, Task scheduling, Makespan, Energy consumption, Multiprocessor, Genetic Algorithm, Energy Conscious Scheduling.

1. INTRODUCTION

The growth of multi-processing applications is developing rapidly and is widely used in different applications, including medical care, smartphones, aerospace, automotive electronics, etc. [1, 2]. To fulfil the necessity of computational abilities in most real-time applications, the processing systems gradually adopt multiple processors for better power saving and energy [3]. For effective performance, proper scheduling is undertaken for dependent or independent tasks for a parallel implementation over a wide range of processors [4]. In computer systems, the multiprocessor scheduling of tasks [5] is one of the main challenging issues. This form of scheduling varies from the conventional scheduling issues as it neglects the task assumption that it can be performed over only one processor in a specified time [6]. Based on the

requirements in a multiprocessor task system, the tasks should be handled on multiple processors simultaneously.

With the developing computational requirement of huge-scale applications, Fog or cloud computing permits a high-velocity placement of huge-scale applications in the present days. The cloud promotes flexible and resistant computing devices for processing diverse tasks. Multiprocessor task scheduling is positioning tasks in a specified manner in a fog or cloud environment to utilize resources appropriately. The fog-cloud computing is a significant technology that renders effective Internet services. Scheduling diverse tasks in cloud computing is not simply due to heterogeneity characteristics. To reduce the overall execution time when following the task dependencies, efficient scheduling and traffic balancing cloud environment should be adopted to perform multi-processing tasks. A fog or cloud environment is a decentralized computing system where data can be stored. A multi-processing environment deals with processing more than one task simultaneously to improve the system's performance.

Task scheduling in computing scenarios is widely classified into dependent and independent strategies [7]. The dependent strategy has communications and dependencies while distributing and allocating tasks over the specified computing resources. The tasks are individually circulated over computing resources, either utilizing a batch or online mode in the case of independent strategies [8, 9]. Specifically, fundamental scheduling issues in multiprocessor systems have become a significant challenge to recent computer advancements [10]. One of the foremost factors that affect a multiprocessor system's performance is improper allocation of tasks, increased makespan and consumption of more energy [11]. The task scheduling problem leads to more time and energy consumption in determining the best schedule.

As the Internet of Things (IoT) based multiprocessor systems are widely used, the total number of connected devices is getting huge, and the data to be processed that emerges from these devices is large [12, 13]. Cloud computing is a significant technology that renders unlimited storage and processing power capabilities. But there are certain issues like increased latency, low performance, bandwidth obstacles and security issues [14]. To attain rapid response and increased quality in scheduling the tasks effectively, cloud-fog computing is utilized as it possesses reduced computational capabilities, and limited resources are required for the task execution [15, 16]. Scheduling all the resources is an essential issue with the multiple connections of devices running various tasks. Enhancing performance and reducing costs when connected to IoT devices are highly challenging.

Multiprocessor scheduling is considered a non-deterministic polynomial-time (NP) hard problem that must be addressed effectively [17]. Normally, the cloud-fog system comprises several computing nodes and upgraded server performance incorporated with many processor cores [18]. The interactions between fog and cloud nodes are relatively complex [19]. For instance, the allocation of resources in handling user tasks is challenging. For the better allocation of resources, the task scheduling approaches are essential in a dynamic environment that assists in establishing the arrival time of each task associated with the resource management system [20]. To conquer the issues of task scheduling, some approaches, including Ant colony optimization (ACO), Particle swarm optimization (PSO) and cuckoo search optimization (CSO), have been established. The approaches are utilized in some deep learning applications, including visual recognition, speech enhancement [21], mood recognition [23,24], emotion recognition [22,25], fraudulent detection, healthcare, etc.

Due to the consumption of more energy, increased makespan, low rate of convergence and computational complexities, better scheduling of tasks cannot be attained in the fog cloud setting. The computation of the multiprocessor system is said to be efficient if it assures better performance in scheduling tasks. Most researchers have worked on multiprocessors' effective task scheduling process recently. Several learning approaches are introduced to attain better

performance. But still, optimal results cannot be obtained due to drawbacks like lower convergence rate, high chances of overfitting issues, high energy consumption and increased makespan. These drawbacks motivated many researchers to promote significant techniques for effectively scheduling tasks. Hence, the Hybrid Genetic Algorithm and Energy Conscious Scheduling approach is implemented to effectively schedule tasks in the proposed work. The proposed task scheduling model is implemented to answer the following questions.

a) How effectively has task scheduling contributed to the proposed research study?
b) What are the scheduling outcomes in the case of multiprocessor systems compared to the existing techniques?
c) Whether the proposed task scheduling algorithms been analyzed statistically?
d) What are the drawbacks analyzed in the case of a proposed task scheduling model?

Some effective contributions in promoting better scheduling tasks to enhance the proposed research's overall network performance are presented below:

- ❖ The transferring and receiving resources in a fog-cloud environment consume more time and energy. To bring out efficient performance, a novel approach called Hybrid Genetic Algorithm and Energy Conscious Scheduling (Hgecs) is presented to minimize the overall energy consumption through efficient scheduling of tasks in the appropriate nodes.
- ❖ To lessen the computational time, energy consumption and makespan issues in the Fog cloud environment by reducing the data traffic with appropriate scheduling to obtain better outcomes for multiprocessor systems.
- ❖ The prioritization of tasks is rendered by implementing the Hybrid Genetic Algorithm, whereas the primary chromosomes are generated. Through the Heuristic model of Energy Conscious Scheduling (Ecs), the tasks assigned to the processor are optimized to promote overall efficiency.

The proposed work is organized into various categories, represented as follows: The associated works of multiprocessor task scheduling in a fog cloud environment are discussed in Section 2. The system architecture of the proposed methodology, problem formulation and model of task scheduling is discussed in Section 3. The outcomes and discussion of the proposed work are provided under Section 4. Lastly, Section 5 concludes the proposed work with future scope and references.

2. RELATED WORKS

Several methods have accomplished optimal results of multiprocessor scheduling tasks in the fog-cloud environment, and some recent related works are described as follows:

A multi-objective method was presented by Abdel et al. [26] based on the Modified sine-cosine Algorithm (MSCA) to tackle the task scheduling process in a multiprocessor. This approach was proposed to improve the makespan and energy by adopting the Pareto dominance strategy. This form can be expanded as an Energy-aware multi-objective MSCA (EA-M2SCA). The classical SCA was modified depending on the optimization process's division into three levels. The first phase identifies a search space during the initial optimization procedure. The second phase explores a randomly chosen population. The third phase searches for the optimal solution for convergence acceleration. To enhance the performance, the proposed approach was integrated with the approach of polynomial mutation. High convergence can be obtained, but the scheduling efficiency is low.

Hassan *et al.* [27] established a significant methodology through which the task priorities were designed from the bottom-level parameter and fuzzy logic. The proposed approach in this research was designed to identify tasks' scheduling by ideal or sub-ideal lengths to obtain high performance across a multiprocessor environment. By adopting an acyclic graph, the precedence constraints were analyzed. As the processor's number was fixed, the

communication cost was insignificant, and the processors were consistent. The proposed technique was established and compared with Prototype Standard Task Graph Set. High performance was attained in the case of makespan, speedup and efficiency. The flexibility and task scheduling capability of the proposed approach were less.

An energy-conscious methodology for multi-core scheduling called a non-pre-emptive dynamic window (NPDW) was introduced by Michel *et al.* [28]. The main aim of this approach was to obtain an effective balance of load and temperature over chip multiprocessors. NPDW utilized the concept of dynamic time windows to accumulate the tasks and determine the best stable match between accumulated tasks and available processor cores by utilizing a modified Gale-shapely approach. The window metrics and match performance were defined to generate a dynamic window to analyze the next time window size. Compared to the baseline schedulers in a multi-core environment, this research distributed the computational and thermal load throughout the processors. Effective system load dispersion and excess core heating can be prevented. The main drawbacks of this research were certain hardware limitations and the window-based scheduling being ineffective.

An efficient energy reduction technique called Dynamic voltage and frequency scaling (DVFS) was contributed by Deng *et al.* [29]. The major objective of this research was to determine the scheduling issue of energy optimization under reliability constraints and makespan on heterogeneous multiprocessor systems through DVFS. An improved WOA (Whale optimization algorithm) was initially proposed by employing an individual selection policy and opposition-based learning. Through this, the ability of exploration and exploitation can be balanced. A constrained approach based on ranks was applied to retaining some infeasible individuals among the population. The performance of improved WOA was enhanced by rescheduling the Critical Path Nodes (CPNs). The redistribution of critical paths enhanced the opposite learning and individual selection policy performance (OIWOA), but an energy consumption problem occurred.

The Energy-Aware Distributed Hybrid Flow Shop Scheduling Problem with Multiprocessor Tasks (EADHFSPMT) was addressed in this research by considering two objectives, including overall energy consumption and makespan. A Novel Multi-Objective Evolutionary Algorithm model based on Decomposition (NMOEA/D) was projected by Jiang *et al.* [30]. Four rules were adopted to begin the population with some diversity based on various relationships between the solution and its neighbours. A cooperative search was designed to generate new resolutions. Two local intensification processes were employed to improve the solution quality. A dynamic adjustment policy was designed to balance convergence and diversity for weight vectors. The overall energy consumption was highly minimized, but the lifetime is negligible.

Abualigah *et al.* [31] proposed a novel hybrid antlion optimization method for resolving the arrangement issues of multi-objective tasks through elite-based differential evolution in the cloud computing environment. The multi-objective nature of the issue is referred to as MALO, which is proposed to reduce makespan and increase resource utilization. The antlion optimization approach was improved by adopting elite-based differential evolution as a local search method to improve the exploration capability and neglect local optima tracking. The experiments were conducted over real and synthetic trace datasets through the cloudsim tool kit. The convergence was faster, and huge scheduling problems were overcome in MALO. Better results were attained for response time, makespan and degree of imbalance, but the overloading capacity is considerably less.

The Enhanced version of the Multi-Verse Optimizer (EMVO) technique was established by Shukri *et al.* [32] to promote an effective task-scheduling process. By varying the number of tasks, the results were obtained to makespan time, throughput and utilization of resources. To lessen the makespan time and increase throughput and resource consumption, the EMVO

approach was employed to schedule the tasks efficiently. The performance of EMVO was compared with the original MVO, and PSO approaches. The outcomes showed that the proposed EMVO in this research had mapped the tasks well by holding no extra overheads regarding throughput. Higher utilization of resources can be attained with less time consumption, but the results of makespan time degraded the system's overall performance.

A multi-unmanned aerial vehicle (multi-UAV) incorporated with a Mobile edge computing (MEC) system was proposed by Luo *et al.* [33]. During offloading tasks over ground users, the UAVs act as computing servers. Through the joint optimization of UAV task scheduling, sharing bits and UAV trajectory in a combined architecture minimizes the energy consumption for ground users. A two-layer optimization policy was introduced, whereas the UAV task scheduling in the upper layer was based on a dynamic program-built bidding optimization approach. The minor layer resolves the bit allocations and trajectory of the UAV. To minimize the computational complexity, the minor layer was separated into various subproblems that can be solved simply by utilizing an alternating multiplier technique, which leads to conflicts. This was eliminated by proposing a re-optimization policy. But the trajectory conflicts cannot be solved effectively.

Aider *et al.* [34] investigated using a look ahead strategy incorporating linking paths to handle the multiprocessor task scheduling issue over two dedicated processors. The major objective of this research was the effective scheduling of tasks to increase network performance. Initially, the proposed method started with an optimal solution constructed by applying the knapsack rule. A sequence of local operators was added to drive the search process around the neighbourhood series. An initial diversification policy was employed based on the drop and rebuild operators. The performance was highlighted through the second diversification policy. The behaviour of the proposed approach was estimated by rendering a statistical analysis through the sign test and Wilcoxon signed-rank test. Tight experimental approximation ratios were obtained, but the system capability is low.

An effective task-scheduling approach based on Particle Swarm Optimization (PSO) was proposed by Agarwal *et al.* [35]. To neglect the premature convergence and convergence acceleration in standard PSO, an opposition-based approach was utilized in this research. The projected approach's performance relates to well-recognized task scheduling policies based on PSO, Modified PSO (MPSO), genetic algorithm and max-min methods. The results obtained after conducting different experiments establish that the proposed scheduling policy based on opposition-based learning-inspired particle swarm optimization performs better with a minimum execution and completion time. The feature consideration of underlying computer resources was low, and the energy consumption was very high.

Hoseiny *et al.* [36] proposed a real-time randomized procedure to conquer the problem of large-scale fog-cloud computing systems. As the obligation for enlightening the quality of service (QoS) and computational power increased intensely, the QoS was raised by adopting the Po2C (Power of Two choices), efficiently reducing the monetary cost. Through the usage of general experiments, the efficacy of Po2C was projected. Improved outcomes were attained compared to the baseline scheduling policies. The deadline violation cost was minimized by 58% compared to the Round robin methodology. The main disadvantage experienced in this research was increased total consumption of energy.

Ali *et al.* [37] established an automated task scheduling method for Fog-cloud systems by adopting a non-dominated sorting genetic algorithm. This research developed a multi-objective task distribution-based optimization method to decrease the overall costs and makespan in fog-cloud computing. For the automatic allocation of tasks, the researchers presented a discrete non-dominated sorting genetic algorithm processed over cloud or fog nodes. The NSGA-II method is applied to work cross-over and mutation processes rather than continuous operators because it consumes huge computational resources and cannot provide suitable task allocation.

This research model professionally shares the workloads through numerous computing fog resources. The simulation results display that the proposed model attains dynamic task scheduling with less execution time. The convergence issue was the major drawback found in this research.

Bacanin *et al.* [38] presented an enhanced firefly optimization algorithm for handling workflow allocations over a cloud-edge environment. Incorporating a quasi-reflection and genetic operator-founded learning approach can overcome the examined complexities of the original firefly algorithm. Initially, the presented approach was validated over ten modern effectual benchmark examples with a performance comparison. Next, the simulations covered workflow scheduling issues by considering makespan and cost. This research can enhance the convergence speed, but established scheduling plans were ineffective.

Chandrashekar *et al.* [39] developed an effective task scheduling procedure that compared cost, makespan and efficiency. The major objective of this research work was to solve the scheduling issues by adopting the Hybrid Weighted Ant Colony Optimization (HWACO) algorithm. Through this research, better convergence can be obtained in a minimal period. The issues of cost and makespan can be overcome, whereas the cloud computing environment performance can be maximized. But the presented research work is highly influenced by greater energy consumption.

Saif *et al.* [40] developed a Multi-Objectives Grey Wolf Optimizer (MGWO) procedure to minimize the major QoS Objectives like energy consumption and delay. Here, the fog broker plays a significant role in disseminating tasks. A mathematical approach of queue theory was employed to allocate workloads effectively to minimize the delay and consumption of power. The stability can be effectively maintained, whereas the major limitation was the non-consideration of resource heterogeneity. Resource utilization and load imbalance were found to be the major issues. Table 1 analyses the task scheduling process in multiprocessor systems with its merits and demerits.

Table 1: Review of task scheduling in multiprocessor systems

Author name	**Techniques used**	**Objective**	**Merits**	**Demerits**
Abdel *et al.* [26]	MSCA	To enhance the makespan and energy by utilizing the Pareto dominance policy.	High convergence can be attained.	Due to the low system efficiency of the Pareto dominance policy, scheduling complex tasks is difficult, which leads to high execution time.
Hassan *et al.* [27]	Performing non-pre-emptive tasks using fuzzy logic.	To determine the scheduling of tasks through best or sub-best lengths in a multiprocessor environment.	High presentation is obtained to makespan, speedup and efficiency.	Optimal scheduling length cannot be assessed due to inefficient fuzzy rules, so the system was not flexible.

Michel *et al.* [28].	NPDW	To achieve an efficient balance of load and temperature on-chip multiprocessors.	Dispersion of system load and excess core heating can be preserved.	The system temperature is maximized when scheduling huge tasks due to limited hardware.
Deng *et al.* [29]	DVFS and OIWOA	To address the scheduling issue of energy optimization on heterogeneous multiprocessor systems.	Enhanced performance of selection strategy.	Because of heavy traffic, energy consumption is more, leading to reliability-aware scheduling issues.
Jiang *et al.* [30]	NMOEA/D	To maintain an effective balance between convergence and diversity.	Energy consumption is highly minimized.	Because of lagging over task balancing capability in NMOEA/D, the system lifetime becomes negligible.
Abualigah *et al.* [31]	MALO	To improve the exploration capability and to avoid local optima tracking.	High convergence, reduced makespan and response time.	Because of huge task sets and less exploration capability, the degree of imbalance and CPU time was very high.
Shukri *et al.* [32]	EMVO	To schedule the tasks effectively and to promote overall performance.	Higher utilization of resources by consuming less time.	The optimization approach converges at higher iterations, whereas the execution time and cost are high over task scheduling.
Luo *et al.* [33]	Multi-UAV with MEC	To lessen the energy consumption of ground users through the joint optimization of UAVs.	Computational complexity is minimized.	During the energy minimization of users through optimization, Trajectory conflicts are

				raised, which cannot be solved effectively.
Aider *et al.* [34]	Look ahead strategy with path relinking.	To schedule the tasks effectively through a diversification policy.	Tight experimental approximation ratios were attained.	Due to the inefficient ranking of tasks, processing tasks are less capable in the overall multiprocessor system.
Agarwal *et al.* [35]	Opposition-dependent learning inspired particle swarm optimization.	To avoid premature convergence for better scheduling of tasks.	Execution and completion time is minimum.	Because of less feature consideration and high energy consumption, the makespan and imbalance degree becomes less in scheduling tasks.
Hoseiny *et al.* [36]	Po2C (Power of Two choices) for diminishing the monetary cost.	To increase the QoS and efficiency of P02C in a fog-cloud computing system.	Cost consumption was greatly minimized.	Because of the huge workload in the network, energy consumption is more.
Ali *et al.* [37]	NSGA-II approach in Fog-cloud systems.	To develop a multi-objective task allocation procedure to minimize energy consumption.	Minimized execution time in task scheduling.	Because of less training ability, convergence issues are more.
Bacanin *et al.* [38]	Genetic operators quasi-reflected Firefly Algorithm (GOQRFA)	To promote better workflow allocations over a cloud-edge environment.	Reduced cost and makespan.	Ineffective generation of scheduling plans.
Chandrashekar *et al.* [39]	HWACO	To solve the scheduling issues with better convergence rates.	Better efficiency over cloud computing can be obtained.	Because of data handling difficulty, increased energy is consumed.
Saif *et al.* [40]	MGWO	To overcome the power consumption	The heterogeneity of	Due to ineffective scheduling

		and delay issues through effective stability maintenance.	resources was not considered.	efficiency, load imbalance issues are generated.

In the existing research on multiprocessor task scheduling in a fog cloud environment, certain limitations emerge, like low scheduling ability, less flexibility, low consideration of features, high energy consumption, and trajectory conflicts. Low scheduling efficiency demonstrates the incompetent scheduling of multi-processing tasks over fog or cloud environments because of heavy data traffic and overloading issues due to a lack of efficient task managing approaches. Also, less overloading capacity, increased makespan, the system's low capability, and decreased network lifetime are other drawbacks that degrade the performance outcomes of task scheduling in a fog cloud environment. To conquer these limitations in existing approaches, a novel methodology called Hgecs is implemented for better scheduling of tasks.

3. PROPOSED METHODOLOGY

The system strategy and the communication among numerous processors included in the scheduling tasks phase, overall system architecture, problem formulation and description of fitness function are described in this section. In this research, the fog broker constructed in the fog layer is assumed to be the major module of the proposed context.

3.1 System architecture

The generalized fog-cloud model comprises end-user devices and routers that commonly process the requests for scheduling tasks. But in the proposed Hgecs model, multiprocessor systems were considered for requesting the server to capture appropriate information. For organizing the corresponding tasks over fog or cloud nodes, a fog broker is the main component for splitting the tasks to be scheduled in the suitable nodes. Efficient scheduling can be undertaken with minimized computational time and energy consumption. The overall system architecture of fog-cloud computing for better task scheduling is illustrated in Figure 1.

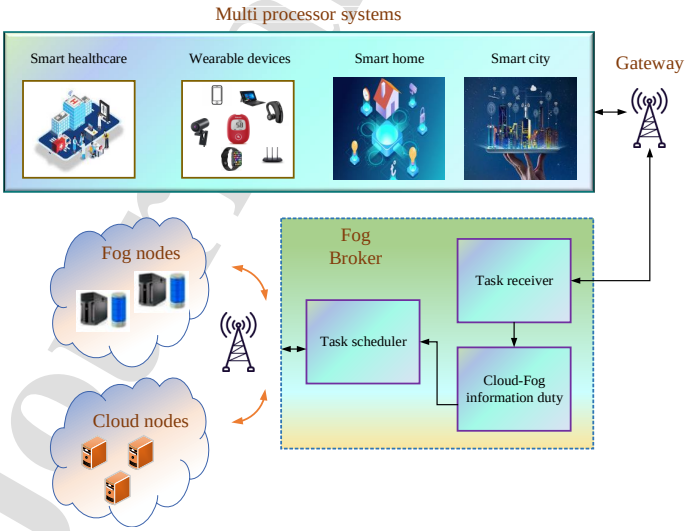


Figure 1: Schematic model of fog-cloud computing

The fog broker comprises three main parts;

- ❖ Task receiver
- ❖ Cloud-Fog information duty
- ❖ Task scheduler

Task receiver

The role of a task assignee is to capture all task requests received from IoT devices and contributors. This component maintains the features and service necessities of the subscribers and then transmits the information to the task scheduler.

Cloud-Fog information duty

The Cloud-Fog information facility receives and monitors the grade reports of possible resources. Also, the status of computing nodes is provided with the task scheduler by this component for better preparation of appropriate scheduling decisions.

Task scheduler

The task scheduler undertakes the task scheduling process by allocating the task appeals to the valuable computing nodes under the task features and capabilities of obtainable resources.

At last, the handled task submissions are gathered and repaid to the fog mediator, which directs the data to respective customers. Figure 2 describes the task scheduling approach between the Fog-cloud system and the fog broker.

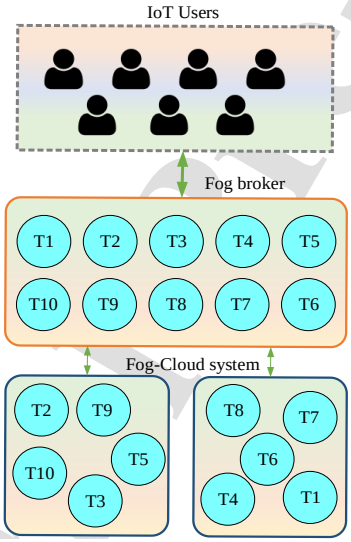


Figure 2: Task scheduling approach

From the figure, it can be demonstrated that the requests are sent from the IoT multiprocessor systems to the fog broker. The tasks are assigned to the corresponding nodes in the fog cloud environment. Traffic may occur during scheduling tasks, leading to increased makespan and energy consumption, whereas proper scheduling cannot be undertaken. The corresponding requests are sent from IoT users to the fog broker. The major function of a fog broker is to split down the vector of separate tasks and creates a Bag of the task list. Based on the scheduling algorithm, the tasks which need to be accomplished are analyzed. The fog broker is a chief concern for transmitting tasks over the corresponding fog and cloud nodes. If the tasks are distributed over cloud nodes, they are forwarded from the cloud system to its corresponding server associated with the cloud node. The tasks are distributed over fog nodes based on the task's priorities. When considering the fog broker assignment, high priority is

given to the fog nodes as they are positioned closer to the users. Lastly, the results from task processing positioned in fog or cloud nodes are transmitted back to the fog broker, and the responses are moved to the respective users. For instance, ten tasks are considered in the above figure and scheduled to the corresponding fog and cloud nodes, respectively. The tasks T_1 to T_{10} are considered, and the fog broker separates them as T_2, T_3, T_5, T_9 and T_{10} are assigned to fog nodes whereas T_1, T_4, T_6, T_7 and T_8 are assigned to the cloud nodes. The task scheduling model's problem description and working using the Hgecs are described in the following sections.

3.2 Problem description

A mathematical description of the problematic scheduling of tasks is described in this section. Consider there are m independent tasks represented as $\left(T=\{T_1,T_2,T_3,....,T_m\}\right)$ where T denotes the task gathered by the fog mediator to be shared through the cloud-cloud and fog-cloud environments. These tasks possess properties, including the file size of input/output, memory requirement, and length of the tasks measured in Millions of Instructions (MI). In accordance with that, the fog-cloud system is assumed to be comprising a set of n computing nodes Com_N which includes n_{cloud} and n_{fog} nodes. Where n_{cloud} and n_{fog} represents the computing nodes of Fog and cloud. The computing node implies $Com_N = n_{cloud} \cup n_{fog}$ and every Com_{Nr}, $r=1,2,3,....n$ holds its characteristics, including the size of the memory, network bandwidth, storage capacity and processing rate of CPU measured in MIPS (Millions of instructions per second). Where Com_{Nr} denotes the computing nodes with physical characteristics, n denotes the final node range and r represents the range of computing nodes.

So, the expected time of computation (ETC) [31] of $P_k, k=1,2,...m$ requests on Com_{Nr}, $r=1,2,3,....n$ is provided by ETC, which P_k denotes the ETC of the task, m represents the number of tasks, n denotes the final node range, r represents the range of computing node and k denotes the time period. The task scheduler utilizes it to evaluate the appropriate scheduling result. Subsequently, the ETC of the task P_k on Com_{Nr} is denoted by $ETC_{k,r}$ which denotes ETC to a period and computing node range. It can be estimated using the below-given equation.

$$ETC_{k,r} = \frac{PL_k}{S_r} \tag{1}$$

From the above equation, the processing speed of Com_{Nr} is denoted as S_r, and the length of the task P_k is represented as PL_k. r represents the range of computing nodes and k denotes the time. The makespan (MS) [41] evaluated for the schedule X is provided in the below equation.

$$MS(X) = \underset{r\in 1,2,...n}{Max} \sum_{k=1}^{m} ETC_{k,r} \tag{2}$$

where, $MS(X)$ denotes makespan for a schedule X, r represents the range of computing node and k denotes the time period and m represents the range setting of time. The energy consumption of the server denotes approximately 60% of its dynamic state [42]. The energy

consumption of Com_{Nr} is denoted by the energy acquired over its constant and active states. The idle time of every Com_{Nr} is represented by its time of execution lessened from its makespan. Hence the energy consumption to Com_{Nr} (calculated in the unit of Joules) can be expressed as follows;

$$E(Com_{Nr}) = (ET_r \times v_r + (MS - ET_r) \times u_r) \times S_r \quad (3)$$

$$v_r = 10^{-8} \times S_r^2 \quad (4)$$

The energy consumed by the cloud node is calculated through equation (4), whereas the energy is 10^{-8} times the processing speed S_r^2 of the cloud node.

$$u_r = 0.6 \times v_r \quad (5)$$

where, $E(Com_{Nr})$ denotes the total energy consumed by the computing node, the total execution time and makespan of corresponding to Com_{Nr} are represented as ET_r and MS. The energy consumed in the active state for Com_{Nr} is represented as v_r, whereas the energy consumed in the passive state for Com_{Nr} is denoted as u_r. The overall energy consumption ($Overall_E$) in the fog-cloud environment can be estimated as follows;

$$Overall_E = \sum_{r=1}^{n} E(Com_{Nr}) \quad (6)$$

where $E(Com_{Nr})$ denotes the total energy consumed and $Overall_E$ depicts the overall energy with a range of the computing node r, and n represents the final range of the node.

3.3 Fitness function

This subsection evaluates the fitness function to identify the outcome quality for the task scheduling problem. The proposed research aims to optimize the entire energy consumption and makespan, as both factors have a major influence over the presentation of a fog-cloud system. This issue is contemplated as a bi-objective issue, whereas the fitness function can be expressed as,

$$F = \delta \times Overall_E + (1 - \delta) \times MS \quad (7)$$

From the above equation, the balance parameter amongst both factors of the corresponding fitness value is represented as δ, whereas 0.7 is set to be the range of δ. $Overall_E$ denotes the overall energy, $(1-\delta)$ denotes the previous balance parameter, MS represents the makespan and F represents the fitness function. Hence, the major aim of the proposed task scheduling approach is to reduce the fitness function. The weighted sum method is commonly adopted to resolve the present bi-objective optimization issue, which includes energy consumption and makespan. The proposed approach can evaluate a single unique result for the tested issue.

4. Task scheduling model using Hgecs

The subscribers of fog-cloud systems do not hold any organization section and can easily acquire the services rendered through the Internet and pay for the utilized services. Such systems can offer any form of services to the corresponding users consisting of web services, computer services, communication and social networking services to the appropriate users. To perform an efficient service, the makespan and energy consumption must be less. A novel methodology called Hgecs is proposed in this research to achieve these objectives. As energy consumption issues tend to degrade the overall task scheduling performance in most previous

works, a hybrid metaheuristic approach is proposed in this research. The genetic algorithm is an efficient task-scheduling approach based on natural genetics and selection. The three basic operations of natural genetics are the selection rule for choosing the individuals, the cross-over operation to integrate two parents for generating a new offspring, and the mutation operation through swapping to attain an optimal output. The genetic algorithm analyses the task priorities, whereas the prioritized tasks are assigned to the appropriate location through the Ecs approach. The Ecs acts as a Makespan-energy minimization approach to exploit indirect energy consumption. The major objective of the Ecs approach is to optimize the output quality of energy and makespan with less complexity over time.

Hgecs integrates the genetic Algorithm and Energy conscious scheduling (Ecs) model. Ecs model is a conscious technique for makespan and energy for effectively scheduling tasks in a fog-cloud environment which involves two steps.

- Every task priority must be accomplished in a specified order, selecting suitable sequences for performing tasks based on priority and adopting a genetic algorithm.
- The Ecs model is adopted in the second phase to optimize the allocated tasks to the processors to utilize resources best and minimize energy consumption.

Hence, the first section of a proposed technique depends on the genetic algorithm, and the major part of a genetic algorithm is to produce the chromosome genes. The genetic algorithm completes the corresponding tasks, $T_1, T_2, T_3, ..., T_m$ and these tasks are based on their genes. The processer and gene voltage part is completed in the next section.

Once the process of a genetic algorithm gets over, the pattern of the Ecs approach is performed. The remaining chromosome parts $C(T_1), C(T_2), C(T_3), ..., C(T_m)$ and voltage parts $V(T_1), V(T_2), V(T_3), ..., V(T_m)$ get completed by Ecs. Where $T_1, T_2, T_3, ..., T_m$ represents the respective tasks which m indicates the total number of independent tasks, C denotes the processors, V describes the voltage. Once the processing procedure of processor and gene voltage is over, the genetic algorithm evaluator function is used. In contrast, part of this function is to evaluate energy utilization and schedule the period of every chromosome. The gene mutation process varies, so the priority is not restricted. The integration of both parents is selected randomly and separated into left and right during practice. The child comprises the right section of two parents.

A genetic algorithm is utilized in the proposed approach to determine allocation results. The proposed algorithm's operators, consisting of chromosome display, initial population generation, cross-over, selection, and mutation process, are clearly described in the upcoming sections. The variables are denoted as constituents called genes in the genetic algorithm approach. The answers to the issues are represented as a chromosome collection of genes. The constituents of the proposed approach are the task program organized on the priority-based chromosome. On every iteration, the gene value can be varied. Also, by applying mutation and cross-over processes, the gene value can be varied again, and new chromosomes (Scheduling) can be gathered. Table 2 represents the chromosome sample regarding eight function programs and three processors.

Table 2: Chromosome sample

T_1	T_2	T_4	T_5	T_3	T_6	T_7	T_8
C_2	C_1	C_2	C_3	C_3	C_1	C_2	C_3
V_3	V_2	V_1	V_1	V_2	V_1	V_2	V_3

The above table illustrates the information regarding the makespan over every processor and the energy consumption. From Table 2, T denotes the task, C represents the processor and V denotes the voltage. For example, in the chromosome mentioned above, T_1 task is performed over the processor C_2 and voltage V_3.

4.1 Population initialization

The initial population selection is made effectively to enhance the speed of tasks and obtain a favourable outcome. By utilizing three prioritization approaches comprising downward priority, upward priority and the integration of these two priorities. The initial three chromosomes are generated, and the three generated chromosomes swap the remaining chromosomes.

Upward priority

The upward priority of the corresponding tasks is equivalent to the remaining average cost until all the tasks regarding the present tasks are finished and must be undertaken after the current task accomplishment. The upward priority is denoted as $Rank_x(T_i)$ and evaluated using the following equation.

$$Rank_x(T_i) = \overline{L(T_i)} + Max\left(\overline{D(T_i + T_j)} + Rank_x(T_j)\right) \quad (8)$$

From the above equation, T_i and T_j denotes the tasks of i, j correspondingly. The weight of each task is represented as $\overline{L(T_i)}$ and $\overline{D(T_i + T_j)}$ denotes the capacity of every task. The performance of the tasks (slower or faster) is shown by $Rank_x$. The successor task sets of T_i are represented as $T_j \in Succ(T_i)$ and $Succ(T_i)$. The upward priority initiates the tasks from the final task T_{Exit} and goes upward.

Downward priority

Comparable to the upward priority, the download task priority is described as $Rank_y(P_i)$ which is evaluated through the following equation.

$$Rank_y(T_i) = Max\left(\overline{L(T_i)} + \overline{D(T_j, T_i)} + Rank_y(T_j)\right) \quad (9)$$

From the above equation, the sums of predecessor tasks of T_i are represented as $T_j \in \Pr ed(T_i)$ and $\Pr ed(T_i)$. $\overline{L(T_i)}$ denotes the weight of each task, $\overline{D(T_j, T_i)}$ denotes the capacity of every task. The download priority can be evaluated by navigating the task graph by initiating the first task. In this approach, the priority of the initial task is 0. $Rank_y(T_i)$ which is equivalent to the largest distance from the initial task to the favourable task (T_i), excluding the computational cost of T_i task.

Combination of priorities

For scheduling directed acyclic graphs, the tasks can be performed based on upward, downward, or combined priorities. In integrating priorities, one level is described for every task based on the following equation, which shows the task level in the graph.

$$LEVEL(T_i) = \begin{cases} 0 & if\ T_i = T_{Entry} \\ Max\left(LEVEL(T_j) + 1\right), & otherwise \end{cases} \quad (10)$$

where $LEVEL(T_i)$ denotes the task level T_i, $Max(LEVEL(T_j)+1)$denotes the maximum level of the task T_j, T_{Entry}represents the entry task and $LEVEL(T_j)$ denotes the task level T_j.

4.2 Selection of parent

In a genetic algorithm, two generations are required to promote the next generation of chromosomes, and the parent's choice depends on various procedures. The algorithm employed in the proposed method to choose the parent is roulette wheel selection. The chromosomes with the best fitness value are chosen; for instance, 20% of the best members from the original population are copied and generate a new population. This way, the members possessing various fitness values have more possibilities. The pairs required to integrate into the next phase are randomly chosen.

4.3 Cross-over process

The information of the two chosen parent chromosomes is combined through the cross-over process and generates the child's chromosomes. Two children can be produced from each combination and select one of these two, which is improved than the parents. The cross-over process in the proposed approach selects two parents randomly, and, in every parent, a random point is chosen, called a cross-over point. The first section of two parents is inserted into two isolated chromosomes. Then the information in the next section will be equivalent to the parent information of the second part. Figure 3 shows the cross-over process between the parent and child.

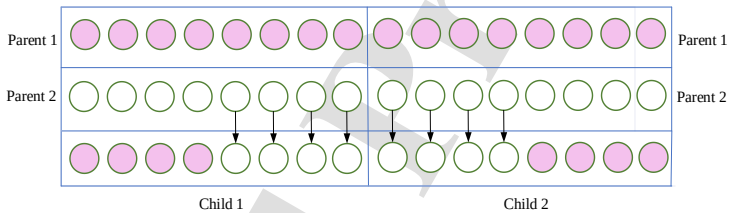


Figure 3: Cross-over operation

Parent 1 and parent 2 are chromosomes separated into left and right portions. The left portion is directly copied over the left portion of chromosomes corresponding to a child that is child 1 and child 2. The second section of the child's chromosome is equivalent to the right portion of parent chromosomes.

4.4 Mutation process

The mutation operator guarantees population diversity by the random variations regarding gene data and intends to preserve the chromosomes from a very high equivalent over various generations. The mutation process in the proposed approach depends upon independent gene displacement at a similar level. The mutation point is chosen randomly, and the initial independent task (gene) swaps the information regarding that gene. It creates a new chromosome, and Figure 4 demonstrates the working process of the mutation.

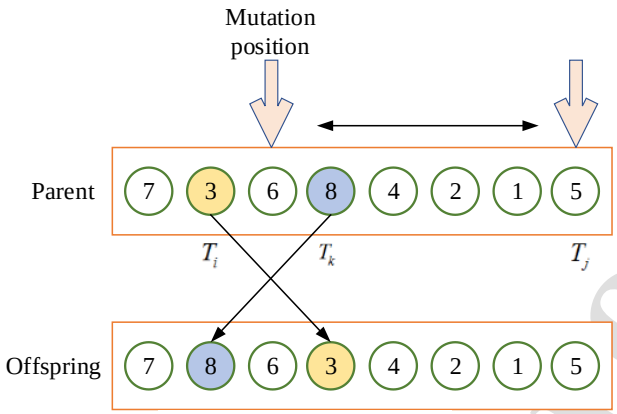


Figure 4: Mutation process

An example of the chromosome is shown in the above figure. The initial T_i substitute from the selected point over the priority queue (T_j) end is determined. Then the exchanging process is done by swapping T_i with the first predecessor T_j called T_k. The illustration of two colours (Yellow and Blue) represents the point of gene exchange.

Ecs technique is an energy and voltage-dependent approach for task scheduling and, at last, allocates to the finest option for processor and voltage. The main aim of using this technique with a genetic algorithm is to schedule the tasks over corresponding processors with minimized makespan and energy consumption criteria. Initially, the tasks are organized in descending order, and for every task, the amount of makespan and energy consumption are evaluated over every processor and voltage. The calculated outcomes are compared with the fitness function. The processors assign the tasks if the value is improved than the preceding states.

When the maximum iteration is attained, the working of a proposed approach is terminated, and the extreme number of iterations considered in this algorithm is 1000.

5. EXPERIMENTAL STUDIES

The proposed Hgecs method is implemented by adopting the MATLAB simulation tool. All the experiments are shown on a Dell PC configured with an Intel (R) Core (TM) i7-3770 CPU of 3.40GHz frequency, 16GB RAM and Windows 7 operating system. The proposed work's fog cloud framework comprises fog nodes with small processing power. They are nearer to the IoT devices, and the delay is minimal. The cloud nodes possess the capability to execute IoT tasks quickly, but they require a long time. Consequently, the proposed algorithm effectively manages the symmetry between the fog and cloud nodes to promote system presentation.

5.1 Evaluation metrics

The main aim of the proposed work is to guarantee less energy consumption and better makespan. The energy consumption and makespan performance are evaluated to calculate the proposed approach's efficiency against the existing methodologies. The two-performance metrics considered are presented as follows:

- The performance metric makespan is the finishing time of the final proficient task. The least makespan insists that effective mapping of user tasks is performed. The computation of the makespan is performed using equation (2).
- The overall energy acquired by physical resources, including fog and cloud nodes, is total energy consumption. In the case of a practical system, the energy consumption of

the computing nodes should be minimum. The overall energy is estimated using equation (6).

5.2 Results and discussion

For the comparative examination, five states of the art procedures are compared with the proposed approach. The existing approaches, including the Gravitational search algorithm (GSA), Round Robin (RR), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and genetic Algorithm (GA), are compared with the proposed approaches to show the proposed performance is better. The outcomes of Hgecs are evaluated for makespan and energy consumption. To promote a better comparison, every algorithm's population size is set to 100. The value δ is set to be 0.7 as the main concern of the proposed work is to lessen the amount of energy consumption.

To review the performance behaviour of the proposed Hgecs method, the curves are plotted for the average fitness values attained by Hgecs against the existing methods of GSA, RR, PSO, ACO and GA by setting a different number of tasks. In this evaluation, the experiments are conducted by considering 300 tasks. The file size is 32 to 64 MB for the task scheduling process, and 50 virtual machines have been used. The capabilities of these virtual machines are illustrated in Table 3.

Table 3: Virtual machine capability

Virtual machine ID (1-10)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Computing power (kbps)	20	25	30	50	55	60	65	66	70	73
Virtual machine ID (11-20)	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Computing power (kbps)	75	80	85	90	35	40	45	50	58	82
Virtual machine ID (21-30)	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Computing power (kbps)	86	91	82	56	38	81	69	86	90	93
Virtual machine ID (31-40)	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
Computing power (kbps)	87	91	91	58	36	54	42	92	89	38
Virtual machine ID (41-50)	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**
Computing power (kbps)	90	87	85	57	69	65	88	96	98	95

The proposed Hgecs possess the ability to resolve the issues properly and can compete with other existing procedures. As Hgecs is a hybrid approach, greater performance regarding makespan and energy consumption can be attained. The hardware limitations of existing approaches can be solved through the proposed research work by overcoming excess power consumption and managing high temperatures through effective task scheduling by avoiding excess data traffic. Also, the processor rates are highly improved for increasing the system performance.

Hgecs holds more ideal conditions to determine makespan, which are measured up to 300 tasks. RR is a scheduling algorithm adopted for the optimal scheduling of tasks, whereas the corresponding job is allocated to a particular time slot. The system ensures scheduling based on priorities by restricting every task over time. But the main drawback faced by the RR algorithm is the reduced processor output because of more time consumption. Based on the swarm behaviour, the PSO algorithm is highly simple to implement and possesses high computational efficiency and robustness in scheduling tasks. Even then, it becomes inferior as it possesses less convergence rate. The ACO approach is used to find the optimal solutions based on the searching behaviour of ants. The time complexity issues can be minimized through this, but the implementation is very hard. The GSA approach works based on newton's law of motion and gravity, which is highly adapted to resolve complex optimization problems. But the efficiency of task scheduling is highly degraded due to local optima problems. A genetic algorithm is an efficient approach requiring only less information for scheduling and is highly probabilistic. But the consumption of energy is found to be more in genetic algorithms. The proposed approach is superior to other existing approaches for accuracy. Figure 5 shows the simulation outcomes of makespan by setting the tasks to 50.

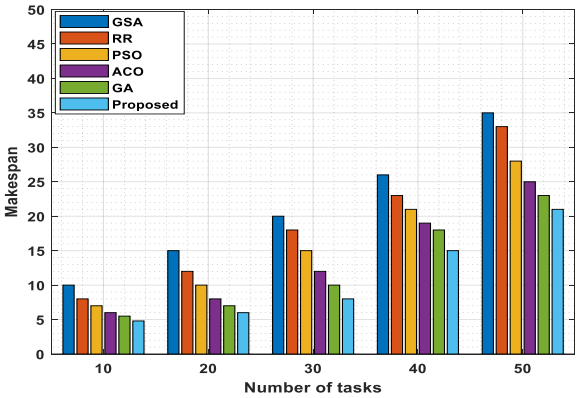


Figure 5: Average makespan for 50 tasks

By varying the tasks from 10 to 50, the makespan performance is evaluated for the proposed and existing approaches. For an effective system, the outcomes of makespan should be minimal, and from the figure, it can be clearly understood that the proposed approach outperforms the existing approaches. The makespan performance is better in the proposed approach by setting the tasks from 10 to 50. In contrast, the makespan performance of the existing approaches like GSA, RR, PSO, ACO and GA did not meet the effective requirements of the system. The existing approaches have attained less scheduling performance due to less convergence rate and increased time and energy complexities. It shows that task scheduling in the proposed approach is done effectively over the resources in the fog cloud environment. Figure 6 illustrates the simulation outcomes of makespan by setting the tasks to 100.

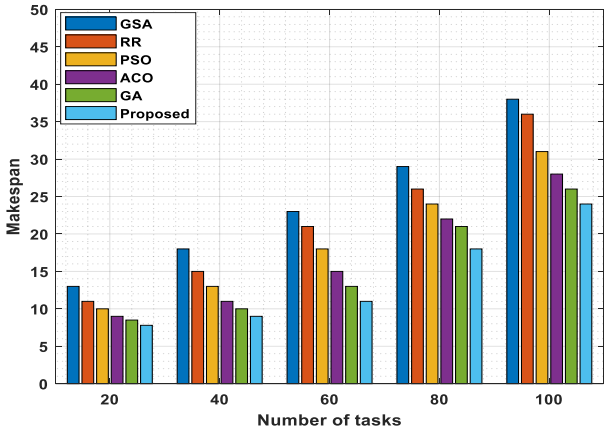


Figure 6: Average makespan for 100 tasks

The proposed algorithm shows better results when the allocated tasks are set to 100 than the existing methods. As the number of tasks gets bigger, the makespan result of different algorithms also increases. The other existing approaches have also attained better outcomes for makespan. Compared to the proposed work, GSA, RR, PSO, ACO, and GA performances are moderately less. Less makespan performance is attained in the existing approaches due to improper scheduling of tasks, increased computation time, and high energy consumption. The existing approaches attain less makespan performance due to improper scheduling of tasks, high congestion, low system efficiency, increased computation time, and high energy consumption. Figure 7 depicts the simulation outcomes of makespan by setting the tasks to 200.

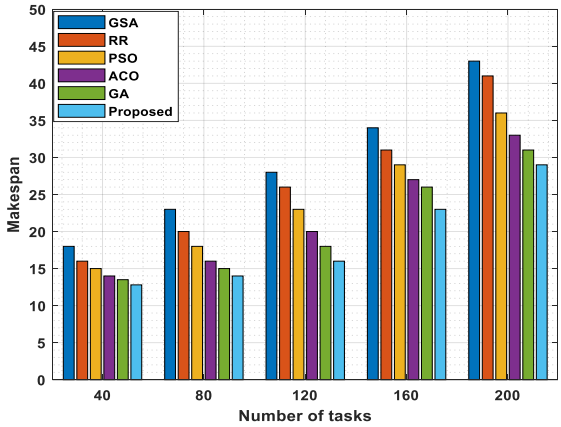


Figure 7: Average makespan for 200 tasks

The average makespan results of the proposed work are superior to existing approaches due to high flexibility, less energy consumption, congestion problem reduction, and increased network lifetime. In the existing approaches like GSA, RR, PSO, ACO and GA, the degraded performance is compared to the proposed method. For tasks in the range of 40, the makespan

performance is comparatively low than the performance achieved with 200 tasks. Figure 8 represents the performance outcomes of makespan by setting the tasks to 300.

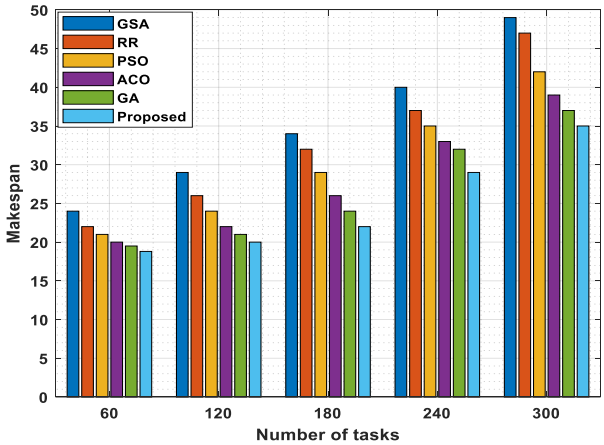


Figure 8: Average makespan for 300 tasks

From the figure, it can be noticed that the proposed work outperforms compared to the existing approaches when the tasks are varied from 60 to 300 tasks. As the overloading capacity of the proposed approach is high, the makespan results are better than the existing techniques like GSA, RR, PSO, ACO and GA. The limitations of the existing approaches are conquered in the proposed Hgecs approach in effectively scheduling tasks to the processors. Figure 9 represents the performance outcomes of energy consumption by setting the tasks to 300.

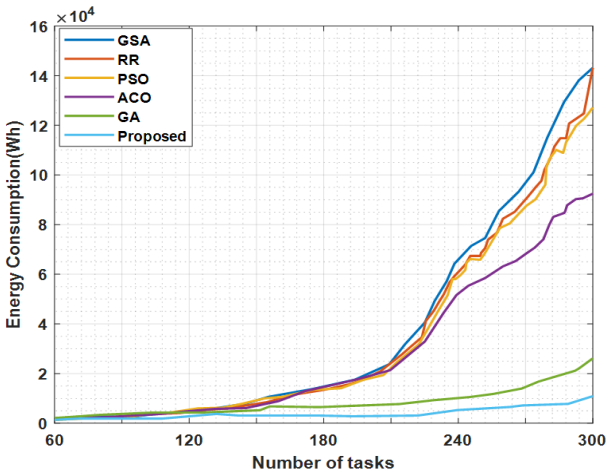


Figure 9: Energy consumption for 300 tasks

The energy results of the proposed approach compared to the existing algorithms are shown in terms of Wh (watt per hour). The performances are estimated for several tasks ranging from 60 to 300. As seen in the figure, energy consumption maximizes when the tasks increase in all the approaches. The Hgecs algorithm promotes better outcomes for determining energy in various tasks. Low energy consumption is obtained due to the lower value of the makespan.

Compared to the existing methodologies like GSA, RR, PSO, ACO, and GA, the proposed approach shows better outcomes due to lower complexities regarding computation and time. Figure 10 illustrates the performance outcomes of fitness values by setting the tasks to 300.

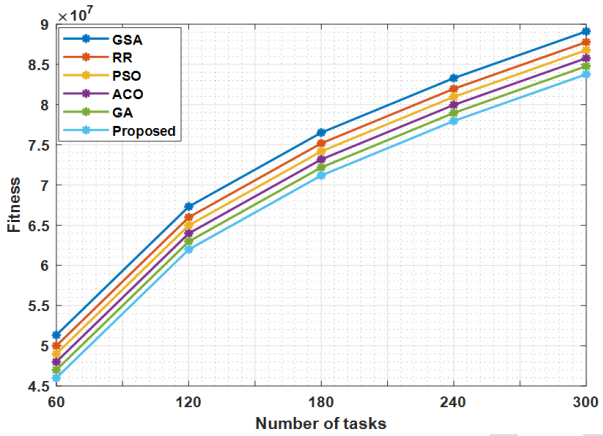


Figure 10: Fitness values for 300 tasks

The above figure shows the fitness curve for the proposed and existing approaches. A clear demonstration can be attained that a lower fitness value is obtained in the proposed Hgecs approach compared to the existing approaches like GSA, RR, PSO, ACO and GA. The task settings are done in the range of 60 to 300. The Hgecs algorithm attains better performance by acquiring lower fitness values other than the existing scheduling approaches, and the proposed method possesses the ability to determine optimal solutions.

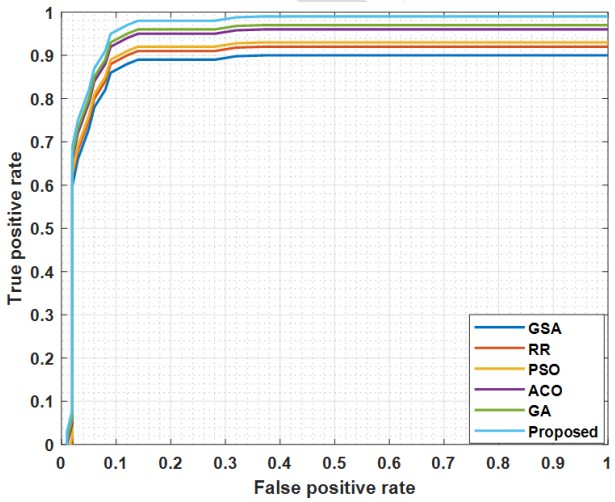
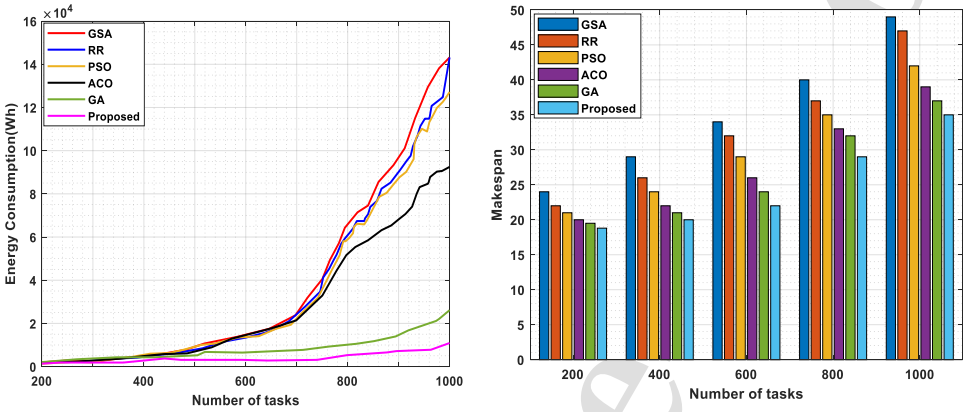


Figure 11: ROC

Figure 11 illustrates the Receiver operating characteristic (ROC) performance analysis of proposed and existing approaches to multiprocessor task scheduling. The ROC is analyzed regarding False positive rate (FPR) and True positive rate (TPR). The optimal cut-off shows the maximum TPR or sensitivity with minimum FPR or specificity. The ROC curves are often

analyzed to show the trade-off between TPR and FPR for every possible scheduling of tasks. It represents the efficiency of task scheduling outcomes with minimizing energy and makespan. The ROC curve also denotes the capability to distinguish the tasks allocated over the corresponding nodes. A higher rate of ROC denotes a better performance of the task scheduling approach. The figure shows that the ROC analysis provided superior values in the proposed model compared with the existing GSA, RR, PSO, ACO and GA techniques. Figure 12 depicts the performance of the makespan and energy consumption for 1000 tasks.



(a) (b)

Figure 12: Scheduling performance for 1000 tasks (a) Energy consumption (b) Makespan

From the figure, it can be justified that task scheduling has attained better results in terms of energy consumption and makespan for 1000 tasks. The tasks are increased from 200 to 1000, and the performance of the proposed scheduling model is analyzed. The existing methodologies like GSA, RR, PSO, GA and ACO are compared with the proposed performance model. Because of higher data traffic and huge time complexity, the existing methods tend to consume more energy and result in increased makespan. Figure 13 shows the Makespan performance of the proposed model in a box plot considering 1000 tasks.

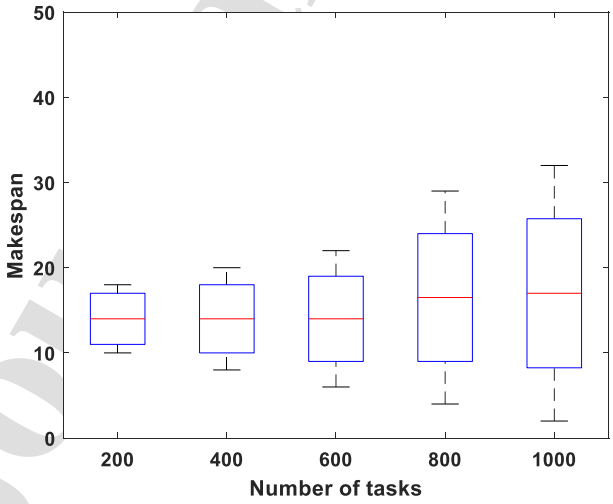


Figure 13: Proposed Makespan performance

When the number of tasks is increased to 1000, the Makespan performance is analyzed in the box plot. The figure's red line indicates the mean value of starting and ending points. It can be observed that the makespan of the proposed model increases when the number of tasks is increased. Compared to existing methodologies, the proposed model attains minimum makespan in scheduling appropriate tasks. Figure 14 shows the convergence plot for metaheuristic optimization approaches.

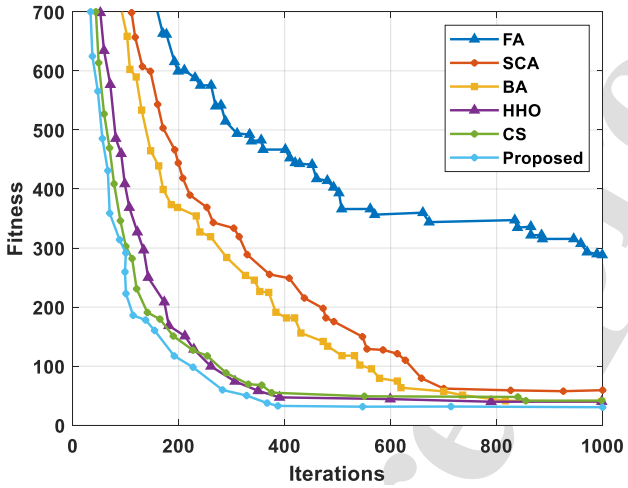


Figure 14: Convergence performance

The convergence performance is analyzed to the number of iterations versus the fitness function of the proposed model. From the figure, a clear analysis can be made that the proposed model converges faster over increasing iterations compared to the existing approaches like the Firefly algorithm (FA), sine cosine algorithm (SCA), Bat algorithm (BA), Harris hawks optimization (HHO) and cuckoo search (CS) optimization algorithm. The existing approaches obtained less convergence rate because of less training ability, high overfitting issues, and increased complexities in time and energy.

5.3 Statistical test analysis

The overall efficiency of the proposed task scheduling model can be estimated by the statistical analysis called the Mann-Whitney U test and post-hoc examination. The Quantitative decision can be rendered through statistical analysis, and from this, it can be observed that the proposed scheduling model promoted improved performance.

5.3.1 Mann-Whitney U test

The task scheduling time of fog and cloud nodes is the variable analyzed in the proposed research work. To estimate the performance of a proposed scheduling model, a statistical non-parametric test called the Mann-Whitney U test is conducted. On conducting the normality test, it can be found that the data are not normally distributed because of uneven task scheduling time. Hence Mann- Whitney U test [43] analysis is carried out in the proposed research work. The Mann-Whitney U test is employed to associate whether there is a difference in task scheduling time for the two independent fog nodes and cloud nodes. The test procures ranking for all the dependent values from low to high. Then the rank sums of task scheduling time corresponding to fog and cloud nodes are evaluated. If the rank sums are similar, it indicates a

null hypothesis that predicts no relationship with the variables. In the proposed research, the rank sums are different and tend to alternate hypotheses. It states that the research procures an effect or relationship for better outcomes. Table 4 shows the outcomes of the Mann-Whitney U test analysis.

Table 4: Mann-Whitney U test analysis

Parameters	**Values**					
	CS	**HHO**	**BA**	**SCA**	**FA**	**Proposed**
Total number of tasks	1000	1000	1000	1000	1000	1000
Mann-Whitney U value	90.68	110.36	542.13	586.97	678.45	88.31
Standard error	11.362	18.154	21.38	22.083	38.657	9.239
z-value	1.990	1.532	1.638	1.286	1.029	1.978
p-value	0.048	0.0453	0.039	1.312	0.986	0.023

Also, the statistical analysis of existing methodologies like FA, SCA, BA, HHO and CS are examined. The proposed work is examined, and the significance level of the non-parametric test is 0.05, which is signified as p value. If the accomplished value is inferior to a certain value, that is $p<0.05$, then the method is considered to be highly significant. If the attained value is higher or equal to the significance level, that is $p\geq 0.05$, then the method cannot render the best performance. The statistical outcome of the Mann-Whitney U test of the proposed task scheduling model is obtained as $p=0.032$. The value of p is inferior to 0.05, and this has been proved statistically that the proposed research work achieves better performance. From the above table, it can be justified that the proposed outcomes are better.

5.3.2 Post-Hoc analysis

The statistical analysis using the Mann-Whitney U test examines the proposed methodology against the existing methods. To determine the time variations of existing algorithms from the proposed model, several post-hoc tests [44] were carried out. The rankings of proposed and existing algorithms with the proposed p-values are presented in Table 5.

Table 5: Average rankings for multiple p-value comparison

Algorithm	**Average rank**	**Proposed p-values**
		Post Hoc Mann-Whitney U test
CS	5.8	0.0462
HHO	5.2	0.0498
BA	3.6	0.0514
SCA	3.5	0.0512
FA	2	0.0491
Proposed	**1**	-

Table 6 shows the post-hoc outcomes for the Mann-Whitney U test for equivalence hypothesis rejection.

Table 6: PostHoc outcomes of Mann-Whitney U test

Algorithms	Z value	P value	Bonferroni	Holm	Holland
CS	18.65	0	0.00	0.011	0.016
HHO	15.54	0	0.00	0.015	0.018
BA	13.85	0	0.00	0.016	0.028
SCA	9.86	0	0.00	0.024	0.053
FA	9.53	0	0.00	0.051	0.052

To reject some hypothesis of equivalence between the effective performing algorithms, the statistical analysis dependent upon the PostHoc algorithms are used by Bonferroni, Holm, Holland, Rom, Finner and Li. In the proposed work, Bonferroni, Holm and Holland procedures are employed. The outcomes of post hoc analysis have been performed with a significance level of 0.05. In the above tables 5 and 6, the rankings are provided for the existing algorithms from the best-performing proposed model. To determine the equivalence hypothesis that can be rejected, the proposed model is compared against certain algorithms. The Z-value and unadjusted p-values are mentioned based on diverse PostHoc procedures. Considering the overall outcomes, the proposed model is proven statistically significant compared to the other algorithms.

6. CONCLUSION

Efficient task scheduling is among the prominent challenges in fog-cloud systems due to the variability, dynamicity of the resources and increased volatility of service requests from the fog-cloud customers. This study presented a hybrid approach integrating GA and Ecs models to determine an optimal solution for effective multiprocessor task scheduling. In most approaches, attention has been provided in case of energy issues. The Hgecs method is utilized in the proposed approach to better schedule tasks that optimize makespan and energy usage. The main focus of a proposed Hgecs procedure is to lessen the makespan and energy consumption. Here, GA generates the best schedules and three ranking approaches to generate the primary chromosomes. The Ecs model is adopted to optimize the resource allocation to the corresponding processors. Using the Hgecs algorithm, three primary chromosomes are generated using three priority methods. The principal chromosomes are fed to GA, and the population is initialized. Better chromosomes are chosen through cross-over and mutation processes. The best chromosomes are selected for the makespan, and the energy consumption is allocated to the processors. The allocation of resources is optimized through the Ecs model. The performance evaluation in MATLAB results in better results in Hgecs than the existing approaches. Even though the proposed work attains better performances, some limitations are noticed. In the proposed model, the global optimum cannot be achieved, and the task scheduling time exists longer due to underrated chromosome generation. Due to some data traffic, energy consumption cannot be reduced at a larger rate. Hence, future work can be prolonged by considering large-scale environments to improve the convergence rate and minimize energy consumption. In contrast, more parameters like response time and transmission costs will be evaluated to analyze the performance of effective task scheduling.

Funding: No funding is provided for the preparation of the manuscript.
Conflict of Interest: The Authors have no conflict of interest to declare.
Data Availability Statement: Data sharing does not apply to this article.

REFERENCES

[1] J. Qiao, H. Wang, & F. Guan, A multiprocessor real-time scheduling embedded testbed based on Linux, International Journal of Embedded Systems. 14(5) (2021) 451-464.

[2] A. Mubeen, M. Ibrahim, N. Bibi, M. Baz, H. Hamam, & O. Cheikhrouhou, Alts: An Adaptive Load Balanced Task Scheduling Approach for Cloud Computing, Processes. 9(9) (2021) 1514.
[3] N. Kumar, J. Mayank, & A. Mondal, Reliability aware energy optimized scheduling of non-preemptive periodic real-time tasks on heterogeneous multiprocessor system, IEEE Transactions on Parallel and Distributed Systems. 31(4) (2019) 871-885.
[4] S. Nabi, M. Ahmad, M. Ibrahim, & H. Hamam, AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing, Sensors. 22(3) (2022) 920.
[5] N. Krishnaraj, & N.B. Prakash, An Intelligent Fitness-Scaling Chaotic Genetic Ant Colony Algorithm Based on Task-Scheduling in Cloud Computing Environments, In Artificial Intelligence Applications for Smart Societies. Springer, Cham (2021) 135-145.
[6] Y.N. Sotskov, & E.I. Mihova, Scheduling Multiprocessor Tasks with Equal Processing Times as a Mixed Graph Coloring Problem, Algorithms. 14(8) (2021) 246.
[7] H. Lee, S. Cho, Y. Jang, J. Lee, & H. Woo, A Global DAG Task Scheduler Using Deep Reinforcement Learning and Graph Convolution Network, IEEE Access. 9 (2021) 158548-158561.
[8] G. Xie, W. Wu, & R. Li, Carry-out Interference Optimization in WCRT Analysis for Global Fixed-Priority Multiprocessor Scheduling, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. (2021).
[9] C. Yang, H. Wang, J. Zhang, & L. Zuo, Semi-partitioned scheduling of mixed-criticality system on multiprocessor platforms, The Journal of Supercomputing. (2021) 1-25.
[10] P.K. Muhuri, & S.K. Biswas, Bayesian optimization algorithm for multi-objective scheduling of time and precedence constrained tasks in heterogeneous multiprocessor systems, Applied Soft Computing. 92 (2020) 106274.
[11] S. Zhao, X. Dai, I. Bate, A. Burns, & W. Chang, DAG scheduling and analysis on multiprocessor systems: Exploitation of parallelism and dependency, In 2020 IEEE Real-Time Systems Symposium (RTSS), IEEE. (2020) 128-140.
[12] Q. Tang, L.H. Zhu, L. Zhou, J. Xiong, & J.B. Wei, Scheduling directed acyclic graphs with optimal duplication strategy on homogeneous multiprocessor systems, Journal of Parallel and Distributed Computing. 138 (2020) 115-127.
[13] M. Kurdi, Ant colony system with a novel Non-DaemonActions procedure for multiprocessor task scheduling in multistage hybrid flow shop, Swarm and evolutionary computation. 44 (2019) 987-1002.
[14] D. Rupanetti, & H. Salamy, Task allocation, migration and scheduling for energy-efficient real-time multiprocessor architectures, Journal of Systems Architecture. 98 (2019) 17-26.
[15] G.L. Stavrinides, & H.D. Karatza, An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations, Future Generation Computer Systems. 96 (2019) 216-226.
[16] J. Cai, R. Zhou, & D. Lei, Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks, Engineering Applications of Artificial Intelligence. 90 (2020) 103540.
[17] S. Kapoor, & SN Panda, Scheduling of Parallel Tasks in Cloud Environment Using DAG MODEL, In Intelligent Computing and Applications. (2021) 267-276. Springer, Singapore.
[18] M. Sulaiman, Z. Halim, M. Waqas, & D. Aydın, A hybrid list-based task scheduling scheme for heterogeneous computing, The Journal of Supercomputing. 77(9) (2021) 10252-10288.
[19] Q. Wu, M. Zhou, & J. Wen, Endpoint communication contention-aware cloud workflow scheduling, IEEE Transactions on Automation Science and Engineering. (2021).

[20] A.A. Alsheikhy, Dynamic approach to minimize overhead and response time in scheduling periodic real-time tasks. (2021)
[21] G. Agarwal, & H. Om, Parallel training models of deep belief network using MapReduce for the classifications of emotions, International Journal of System Assurance Engineering and Management. (2021) 1-16. https://doi.org/10.1007/s13198-021-01394-3
[22] G. Agarwal, V. Maheshkar, S. Maheshkar, & S. Gupta, Vocal Mood Recognition: Text Dependent Sequential and Parallel Approach, In Applications of Artificial Intelligence Techniques in Engineering. (2019) 131-142. Springer, Singapore. DOI:10.1007/978-981-13-1819-1_14.
[23] G. Agarwal and H. Om, An efficient supervised framework for music mood recognition using autoencoder-based optimized support vector regression model. IET Signal Processing 15(2) (2021) 98-121.
[24] G. Agarwal, H. Om and S. Gupta, A learning framework of modified deep recurrent neural network for classification and recognition of voice mood. International Journal of Adaptive Control and Signal Processing. (2022)
[25] G. Agarwal and H. Om, Performance of deer hunting optimization based deep learning algorithm for speech emotion recognition. Multimedia Tools and Applications 80(7) (2021) 9961-9992.
[26] M. Abdel-Basset, R. Mohamed, M. Abouhawwash, R.K. Chakrabortty, & M.J. Ryan, EA-MSCA: An effective energy-aware multi-objective modified sine-cosine algorithm for real-time task scheduling in multiprocessor systems: Methods and analysis, Expert systems with applications. 173 (2021) 114699.
[27] H.E. Hassan, G. Nagib, & K.H. Ibrahiem, A novel task scheduling approach for dependent non-preemptive tasks using fuzzy logic, IET Computers & Digital Techniques. 15(3) (2021) 214-222.
[28] M. Michel, & H. Lee, Energy Conscious Dynamic Window Scheduling of Chip Multiprocessors, (2022). arXiv preprint arXiv:2202.06422.
[29] Z. Deng, D. Cao, H. Shen, Z. Yan, & H. Huang, Reliability-aware task scheduling for energy efficiency on heterogeneous multiprocessor systems, The Journal of Supercomputing. 77(10) (2021) 11643-11681.
[30] E. Jiang, L. Wang, and W. Jingjing, Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks, Tsinghua Science and Technology. 26(5) (2021) 646-663.
[31] L. Abualigah, & A. Diabat, A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments, Cluster Computing. 24(1) (2021) 205-223.
[32] S.E. Shukri, R. Al-Sayyed, A. Hudaib, & S. Mirjalili, Enhanced multi-verse optimizer for task scheduling in cloud computing environments, Expert Systems with Applications. 168 (2021) 114230.
[33] Y. Luo, W. Ding, & B. Zhang, Optimization of task scheduling and dynamic service strategy for multi-UAV-enabled mobile-edge computing system, IEEE Transactions on Cognitive Communications and Networking. 7(3) (2021) 970-984.
[34] M. Aïder, F.Z. Baatout, & M. Hifi, A look-ahead strategy-based method for scheduling multiprocessor tasks on two dedicated processors, Computers & Industrial Engineering. 158 (2021) 107388.
[35] M. Agarwal, & G.M.S. Srivastava, Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing, Journal of Ambient Intelligence and Humanized Computing. 12(10) (2021) 9855-9875.

[36] F. Hoseiny, S. Azizi, M. Shojafar and R. Tafazolli, Joint QoS- aware and Cost-efficient Task Scheduling for Fog-Cloud Resources in a Volunteer Computing System (2021) arXiv preprint arXiv:2104.13974.
[37] I.M. Ali, K.M. Sallam, N. Moustafa, R. Chakraborty, M.J. Ryan and K-K.R. Choo, An automated task scheduling model using non-dominated sorting genetic Algorithm ii for fog-cloud systems. IEEE Transactions on Cloud Computing (2020).
[38] N. Bacanin, M. Zivkovic, T. Bezdan, K. Venkatachalam and M. Abouhawwash, Modified firefly algorithm for workflow scheduling in cloud-edge environment. Neural computing and applications 34(11) (2022) 9043-9068.
[39] C. Chandrashekar, P. Krishnadoss, V.K. Poornachary, B. Ananthakrishnan and K. Rangasamy, HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing. Applied Sciences 13(6) (2023) 3433.
[40] F.A. Saif, R. Latip, Z.M. Hanapi and K. Shafinah, Multi-objective Grey Wolf Optimizer Algorithm for Task Scheduling in Cloud-Fog Computing. IEEE Access (2023).
[41] M. Lavanya, B. Shanthi and S. Saravanan, Multi objective task scheduling algorithm based on SLA and processing time suitable for cloud environment. Computer Communications 151 (2020) 183-195.
[42] M.A. Elaziz, L. Abualigah, R.A. Ibrahim and I. Attiya, IoT workflow scheduling using intelligent arithmetic optimization algorithm in fog computing. Computational intelligence and neuroscience 2021 (2021).
[43] U.O. Eric, M.O. Olusola and P.A. Esemokumo, Statistical Analysis of the Median Test and the Mann-Whitney U Test. International Journal of Advanced Academic Research 7(9) (2021) 44-51.
[44] D.G. Pereira, A. Afonso and F.M. Medeiros, Overview of Friedman's test and post-hoc analysis. Communications in Statistics-Simulation and Computation 44(10) (2015) 2636-2653.

Credit Author Statement

1. **Dr. Sachi Gupta:** Writing - Original Draft Preparation, specifically writing the initial draft.
2. **Dr. Rakesh Ahuja**: Writing - Review & Editing Preparation, specifically critical review, commentary.
3. **Mr. Gaurav Agarwal:** Writing - Review & Editing Preparation, specifically critical review, commentary.
4. **Dr. Atul Kumar Rai:** Writing - Review & Editing Preparation, specifically critical review, commentary.

Declaration of Interest Statement:

Authors Dr. Sachi Gupta, Dr. Rakesh Ahuja, Gaurav Agarwal, Dr. Atul Kumar Rai, declares that they have no conflict of interest. Patients' rights and animal protection statements: This research article does not contain any studies with human or animal subjects.