Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Improved Jellyfish Algorithm-based multi-aspect task scheduling model for IoT tasks over fog integrated cloud environment

Nupur Jangu and Zahid Raza*

Abstract

Corporations and enterprises creating IoT-based systems frequently use fog computing integrated with cloud computing to harness the benefits offered by both. These computing paradigms use virtualization and a pay-as-you-go strategy to provide IT resources, including CPU, memory, network and storage. Resource management in such a hybrid environment becomes a challenging task. This problem is exacerbated in the IoT environment, as it generates deadline-driven and heterogeneous data demanding real-time processing. This work proposes an efficient two-step scheduling algorithm comprising a Bi-factor classification task phase based on deadline and priority and a scheduling phase using an enhanced artificial Jellyfish Search Optimizer (JS) proposed as an Improved Jellyfish Algorithm (IJFA). The model considers a variety of cloud and fog resource parameters, including speed, capacity, task size, number of tasks, and number of virtual machines for resource provisioning in a fog integrated cloud environment. The model has been tested for the real-time task scenario with the number of tasks considering both the smaller workload and the relatively higher workload scenario matching the real-time situation. The model addresses the Quality of Service (QoS) parameters of minimizing the batch's make-span time, lowering the batch execution costs, and increasing the resource utilization. Simulation results prove the effectiveness of the proposed model.

Keywords: Cloud computing, Fog computing, Fog integrated cloud, Resource provisioning, Task scheduling, Metaheuristics

Introduction

In recent decades, the scientific community has embraced meta-heuristic optimization approaches to solve complicated optimization problems. Neural networks, data mining, industrial, mechanical, electrical, software engineering, and specific issues in location theory are some of the application domains of meta-heuristic algorithms [1–5]. Hussain's analysis of 1,222 papers on metaheuristics from 1983 to 2016 (33 years) suggests that the behaviour of birds, humans, plants, water, the ecosystem, electromagnetic forces, and gravitation have been employed as metaphors in metaheuristic techniques [6]. Figure 1 presents a division of these techniques into two groups. The first group includes approaches that imitate biological or physical events and can be divided into four sub-categories: Nature-based, Physics-based, Human-based and Swarm-based methods. The third group consists of those that have been motivated by human events. The most exciting and widely used metaheuristic algorithms are swarm-intelligence algorithms based on the collective intelligence of colonies of ants, termites, bees, flocks of birds, and so on [7]. Their success can be attributed to the fact that they leverage shared knowledge among several agents, allowing self-organization, co-evolution and learning to aid in creating high-quality products during cycles. Although not all swarm-intelligence algorithms succeed, a handful has been quite effective

*Correspondence: zahidraza75@gmail.com
School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi 110067, India

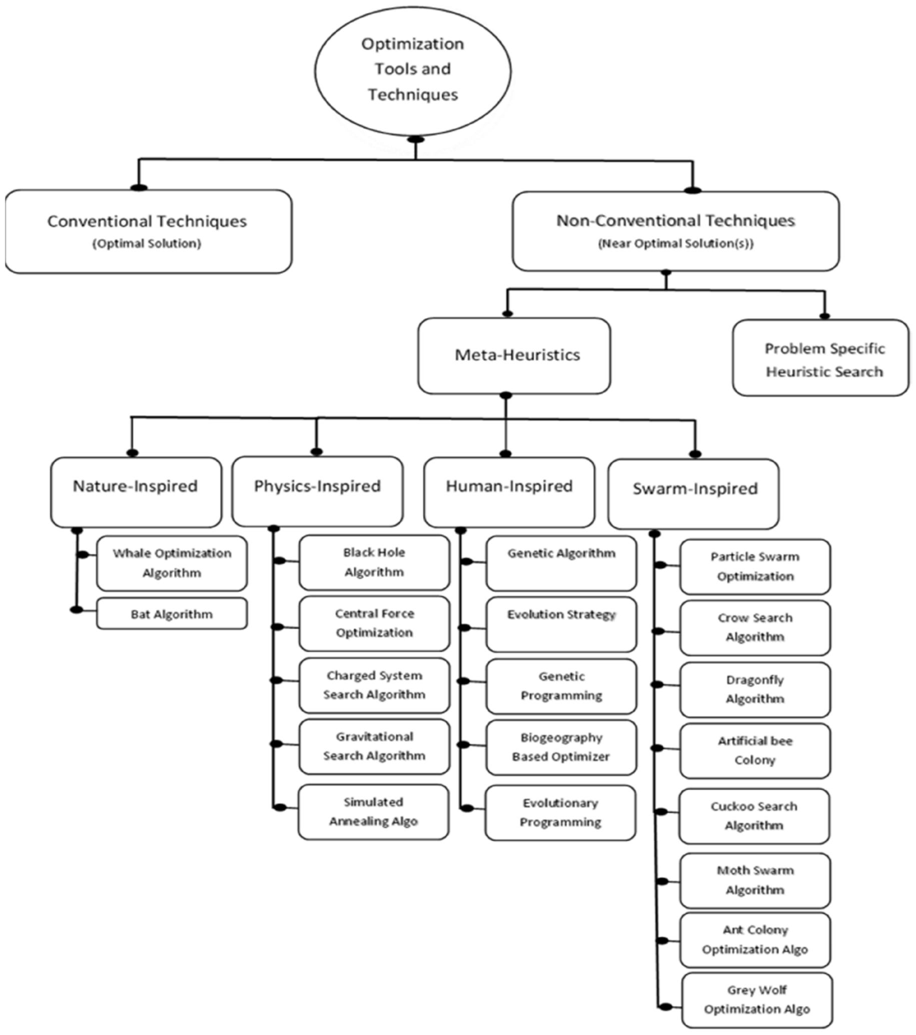
Optimization Tools and Techniques
Conventional Techniques (Optimal Solution)
Non-Conventional Techniques (Near Optimal Solution(s))
Meta-Heuristics
Problem Specific Heuristic Search
Nature-Inspired
Physics-Inspired
Human-Inspired
Swarm-Inspired
Whale Optimization Algorithm
Bat Algorithm
Black Hole Algorithm
Central Force Optimization
Charged System Search Algorithm
Gravitational Search Algorithm
Simulated Annealing Algo
Genetic Algorithm
Evolution Strategy
Genetic Programming
Biogeography Based Optimizer
Evolutionary Programming
Particle Swarm Optimization
Crow Search Algorithm
Dragonfly Algorithm
Artificial bee Colony
Cuckoo Search Algorithm
Moth Swarm Algorithm
Ant Colony Optimization Algo
Grey Wolf Optimization Algo

Fig. 1 Classification of Meta-Heuristic Algorithms

and have thus become popular tools for tackling real-world problems [8].

Working in the same trend, cloud computing research has also leveraged the benefits of many meta-heuristics to target its' complex problems, e.g., virtual machine allocation [9–11], virtual machine placement [12, 13], load balancing [14], task scheduling [15–27], workload prediction [28], resource allocation [29, 30], workflow scheduling [31, 32], virtual machine migration [33] and many more.

Computing paradigms like cloud and related technologies, fog and edge computing are built on the pay-as-you-go model. Resources are provided based on the service-level agreements (SLA) between the service providers and the consumers. Resources are of utmost importance for these technologies. Accordingly, resource provisioning for task scheduling becomes one of the significant concerns for these paradigms alongwith other challenges like security, performance, resource management, reliability etc. Therefore, to achieve an efficient performance and to make the best use of the scarce fog or cloud resources, the users' tasks must be scheduled intelligently on the available resources while meeting the desired QoS. There are numerous factors to be considered for designing any task-scheduling algorithm. Some of critical factors are task completion time, makespan, security, and response time from a user's standpoint. From the service provider's Provider's standpoint, the crucial parameters considered are resource utilization, fault tolerance, and power consumption to name a few.

Owing to the large solution space and time required to obtain an optimal solution, job scheduling, also known as resource provisioning for the cloud and peers, has been classed as NP-hard [34]. Optimization methodologies using meta-heuristic methods based on performance factors, e.g., completion time, cost, resource utilization offer a solution to address the resource provisioning problem. Although cloud computing meets the criteria of executing IoT operations locally, it impacts task performance and requires a platform to handle these tasks. Furthermore, IoT applications conduct various tasks with varying priorities and deadlines that must be performed without delay. Still, remote execution of these tasks in the cloud server creates multiple issues, including higher latency and limited bandwidth availability. These challenges can be overcome by processing IoT tasks at the network's edge, called fog computing [35]. The goal of fog computing is to complete work before the deadline with local workload execution. The hybrid fog-cloud architecture offers a promising arrangement to improve the QoS with a broaded horizon. The fog layer assists the cloud layer in the task execution. Some appropriate tasks are executed at the fog level, with the remainder of the workload offloading to the cloud. Scheduling tasks in the fog-cloud layers promise to achieve reduced make-span time in executing these tasks with a better resource utilization.

Looking at the immense benefits of using meta-heuristics in computing literature, this study proposes an Improved Jellyfish algorithm (IJFA) model for scheduling classified tasks over the Fog integrated cloud environment. The work considers the tasks originating from the Internet of Things (IoT) devices, resulting in a heterogeneous task generation. The model uses a bi-factor classification method based on task category considering the priority and deadline, and the resource requirements aiming to minimize the make-span time of the batch of jobs. This, in turn, will lower the execution cost of tasks submitted for execution while improving the overall resource efficiency and utilization.

The proposed algorithm IJFA is based on a recently developed population-based nature-inspired meta-heuristic algorithm, an artificial Jellyfish Search Optimizer (JS), inspired by the behaviour of Jellyfish in the ocean. The simulation of jellyfish search behaviour includes their following of the ocean current, their motion within a jellyfish swarm as active and passive motions, a temporal control system for switching between these movements, and their convergence into a jellyfish bloom. On benchmark functions and optimization problems, the new approach performes admirably. The population size and the number of iterations are the only two control parameters in JS. As a result, it requires minimum effors in deployment and could be an excellent meta-heuristic algorithm for addressing optimization problems. This work modifies the the JS algorithm in order to to achieve faster convergence by improving its exploration phase resulting in a wider exploration of the solution space to attain efficient scheduling decisions [36].

The remaining section of this research is systematized as follows. Literature survey section provides the literature survey of the various works reported in the literature in the domain. The proposed multi-aspect task scheduling approach section details the proposed model including Improved Jellyfish algorithm (IJFA) based on task scheduling principles. Experimental results section discusses the simulation setup, simulation results and a performance evaluation of the proposed model. Conclusion and future works section 5 presents the conclusion drawn from the work and possible future directions.

Literature survey

Cloud computing is the most popular distributed computing paradigm that provides self-service, dynamically scaled and metered access to a shared pool of resources with guaranteed Quality of Service (QoS) to the users. The jobs must be efficiently mapped to the offered resources to achieve QoS. Otherwise, it may violate Service Level Agreements (SLA). As a result, users will be hesitant to pay if the desired performance is not realized. Therefore, cloud computing systems consider scheduling a significant theme, where obtaining a subpar solution in a short period is desirable. No algorithms can solve the scheduling issue in polynomial time and provide optimal results owing to the vast search space in the actual implementation of the computing world. Therefore, combining meta-heuristic algorithms with the optimization of essential parameters reduces search space complexity and execution time. Also, the goal of task scheduling changes from one application to the next according to the QoS standard requirements. As a result, numerous studies in cloud and fog computing focus on meta-heuristics-based job scheduling. This section provides a comprehensive review of several scheduling techniques using various metaheuristics in the realm of cloud and fog computing.

Meta-heuristics in cloud computing

When implementing a task scheduling approach, at least one objective function ensures high performance. The most prevalent objectives are make-span, monetary cost, computational cost (i.e., CPU, memory, storage, GPU, bandwidth, etc.), reliability and availability, elasticity or scalability, energy consumption, security, resource usage, and throughput [37]. Researchers have explored these single-objective and multiobjective areas interestingly using various metaheuristics.

In [38], the authors introduced a new single objective strategy based on the Firefly algorithm for scheduling submitted tasks in clouds to reduce make-span. Their proposed Firefly Algorithm (FFA) outperforms Simulated Annealing (SA) and the Cuckoo Search Algorithm (CSA) in the experiments. Though this work minimizes the make-span successfully, other parameters, such as monetary cost, scalability, and availability, were ignored. The work in [22] suggested a hybrid task scheduling technique MSA using two metaheuristics MSDE (Hybrid Moth Search Algorithm and Differential Evolution), which is an integration of the Moth Search Algorithm (MSA) and Differential Evolution (DE) algorithms with a single goal of minimizing the make-span time required. The model offers exploration and exploitation capabilities based on Lévy's flight and phototaxis ideas. However, since MSA's exploitation capability is restricted, the DE algorithm has been utilized for local search, offering superior exploitation capability. Their results show that the proposed hybrid MSDE algorithm outperforms state-of-the-art heuristic and meta-heuristic scheduling algorithms regarding system make-span and throughput. Likewise, in [39], the authors proposed a hybrid method that combines the benefits of Ant Colony Optimization (ACO) and Cuckoo search that tries to lower the make-span or completion time. The work achieved this objective using the hybrid algorithm because the jobs were completed by allocating sufficient resources inside the set time interval. The findings suggest that the Hybrid algorithm outperforms the ACO method in terms of algorithm performance and time to completion.

An improved ant colony algorithm for multiobjective optimization scheduling based on a resource cost model (relationship between the user's resource costs and the budget costs) was reported in [40]. The model achieved multiobjective performance and price optimization by including the make-span and the user's budget costs as optimization constraints. Their multiobjective optimization method performed better than similar methods based on the make-span, cost, deadline violation rate and resource utilization. Using the benefits like the speed and accuracy of the PSO algorithm, the authors in [41] proposed a comprehensive multiobjective model to give better QoS to Cloud customers by reducing the task execution/transferring time and cost. The authors tried to achieve it by moving extra tasks from an overloaded VM rather than migrating the complete overload and eliminating the use of the VM pre-copy process. The simulation results reveal that the suggested method dramatically decreases the load balancing time compared to standard load balancing methodologies.

Similarly, research in [42] proposes a PSO-based Adaptive Multiobjective Task Scheduling (AMTS) strategy that considers both processing and transmission time and produces a better quasi-optimal solution in terms of average cost, job completion time and energy consumption, according to experimental results. The work reported using an adaptive acceleration coefficient to preserve particle diversity. Following the same trend, [43] also sed a load-balanced scheduling strategy based on the New Particle Swarm Optimization (NPSO) method. A new cost assessment function was employed to reduce the monetary cost of processing tasks on VMs. The suggested method improves efficiency through cost optimization (minimized cost) based on a statistical analysis of the total cost (execution and transfer) on a data set with many iterations and particles.

To build a complete multiobjective optimization model for task scheduling, the authors in [44] included four conflicting objectives: task transfer time, task execution cost, power consumption, and task queue length to lower expenses for customers and providers. Using the Multiobjective Particle Swarm Optimization (MOPSO) and Multiobjective Genetic Algorithm (MOGA), their proposed multiobjective model achieves optimal trade-off solutions among the four conflicting objectives, reducing job response time and make-span significantly. Findings say that the proposed model is faster and more accurate, improving QoS and lowering provider costs. Authors in [45] introduced a novel Multi-objective Cuckoo Search Optimization (MOCSO) technique for dealing with the resource scheduling problem in cloud computing to lower cloud user costs and improve performance by reducing make-span time. This helps cloud providers earn revenue or profit by maximizing utilization. The investigations and evaluation of the proposed method show that it outperforms MOACO, MOGA, MOMM, and MOPSO in balancing numerous objectives such as projected time to completion and cost. [46] proposed an ACO, PSO, and GA-based task-level and service-level dynamic resource scheduling technique, in which a task is assigned to a VM and a task is assigned to a service, respectively. This solution optimizes the make-span and CPU time while also lowering the overall operational cost of data centres. Still, the model does not perform well when allocating resources to global tasks. Table 1 summarises some of the recent single objective and multiobjective meta-heuristic algorithms employed in the cloud to address various QoS parameters.

Meta-heuristics in fog integrated cloud computing

Taking inspiration from the usage of metaheuristics in fellow cloud computing, researchers have explored the aspect of task scheduling using the same in fog computing and fog integrated cloud environments. In the work [47], the authors offered an energy-saving method based on a meta-heuristic known as the Harris Hawks optimization technique to increase QoS while main-

Table 1 Meta-Heuristics used in cloud computing

	Research	Optimization technique used	Objective
Single Objective	[38]	Firefly algorithm	Reduce make-span
	[22]	Moth Search Algorithm (MSA) and Differential Evolution (DE) algorithms	
Multiobjective	[39]	ACO (Ant Colony algorithm) and Cuckoo search	
	[40]	Ant Colony algorithm	Performance and Cost
	[41]	PSO (Particle Swarm Optimization)	Load balancing and better QoS
	[42]		Best resource utilization, Average cost, the average time to complete a task, and average energy usage
	[43]		Cost Minimisation
	[44]	MOPSO (Multi-Objective Particle Swarm Optimization) and MOGA (Multiobjective Genetic Algorithm)	Reducing job response time and make-span, Provider and consumer cost
	[45]	MOCSO (Multi-objective Cuckoo Search Optimization)	Reducing Make-span
	[46]	ACO, PSO, and GA	Optimizes the make-span and CPU time and lowers the overall operational cost

taining SLA. The proposed algorithm is based on the fact that task scheduling is essential and adds to fog servers' energy usage when managing Industrial IoT (IIOT) applications. It reportedly beats other known algorithms such as Particle Swarm Optimization (PSO) and Teaching Learning Based Optimization (TLBO) while considering the performance in terms of energy consumption and other QoS factors. In [48], the authors introduced an Adaptive Double fitness Genetic Task Scheduling (ADGTS) algorithm to maximize task make-span and communication cost at the same time using collaborative task and fog resource scheduling. Simulation results suggest that the ADGTS algorithm can simultaneously balance communication cost and task make-span performance. It performs better considering task make-span than the Min–Min method. Authors in [49] proposed a task scheduling algorithm based on a Moth-Flame Optimization (TS-MFO) algorithm. The proposed technique assigns an appropriate set of tasks to fog nodes to meet the quality-of-service criteria of Cyber-Physical Systems (CPS) applications while minimizing task execution time. The simulation study suggests the outperformance of the model over the PSO, NSGA-II, and (Bees Life Algorithm) BLA techniques in terms of total task execution time. The authors in [50], provides an optimization technique for IoT based applications using modified version of genetic algorithms with a focus on reducing latencies.

On the other hand, few works tried to explore multi-objective scenarios. In one such work [51], the authors tried to improve the QoS supplied to users in (Industrial IoT) IIoT applications to propose an energy-aware meta-heuristic based on a Harris Hawks Optimisation algorithm based on a Local search Strategy (HHOLS) for Task Scheduling in Fog Computing (TSFC) aided by the normalizing and scaling phase in solving the discrete TSFC. The quality of the solution was improved even more by balancing workloads across all virtual machines due to the swap mutation. The work compared their HHOLS method with other meta-heuristics based on various performance indicators, such as energy consumption, make-span, cost, flow time, and emission rate of $CO_{2.}$ Using the same meta-heuristic, the authors in [52] proposed an enhanced elitism genetic algorithm (IEGA) to solve the work scheduling problem for FC and increase the quality of services provided to IoT device consumers. The proposed method demonstrates superior performance in make-span, flow time, fitness function, carbon dioxide emission rate, and energy consumption compared to other peers. The benefits of IEGA come from two main phases: first, the mutation rate and the crossover rate being manipulated to aid the algorithms in exploring the majority of the possible combinations that could form the near-optimal permutation; and second, several solutions being mutated based on a certain probability to avoid becoming trapped in local minima and to find a better solution.

Another work reported in [53] demonstrates a novel scheduling method based on the ant colony algorithm allowing for more accurate job scheduling and execution. It is a three-step method in which tasks are separated into two groups based on their completion time and cost, followed by prioritization based on completion time and cost. Then, the ant colony method is utilized to choose the best virtual computer to run the jobs. Simulation results suggest that the proposed method provides acceptable performance in make-span, response time, and energy usage compared to others. The work in [54] presents a Novel Bio-Inspired Hybrid Algorithm (NBIHA), a mix of Modified Particle Swarm

Optimisation (MPSO) and Modified Cat Swarm Optimisation (MCSO) to lower average reaction time and optimize the resource consumption by efficiently scheduling jobs and managing available fog resources. The proposed algorithm outperforms three other algorithms viz by minimizing the execution time. First Come First Serve (FCFS), Shortest Job First (SJF), and MPSO. In the work [55], the authors proposed a Priority-aware Genetic Algorithm (PGA) which is a hybrid technique combining task prioritization with a genetic algorithm implementation. The objective is to determine the best compute node for each task by considering various job requirements while considering the diverse nature of fog and cloud nodes. It is a novel fog-cloud scheduling algorithm that optimizes a multiobjective function, a weighted sum of overall computation time, energy consumption, and Percentage of Deadline Satisfied Tasks (PDST). The work [56] introduced a Discrete Non-Dominated Sorting Genetic Algorithm II (DNSGA-II) based optimization model to schedule tasks dynamically aiming to reduce the makespan and costs in a fog-cloud environment. The model deals with the discrete multiobjective scheduling problem, allocates computing resources either on cloud or fog nodes and organizes the distribution of workloads. Another work in [57] attempts to reduce the makespan and energy consumption in the fog-cloud environment by using an integration of the Cultural Evolution Algorithm (CEA) and the Invasive Weed Optimization (IWO) named as hybrid (IWO-CA). The Dynamic Voltage and Frequency Scaling (DVFS) technique is presented to minimize the consumption of energy. Whale Optimization algorithm in smart healthcare application has been explored in [58] to address the massive data generated in the process aiming to minimize the average energy consumption and cost. Table 2 summarises some of the recent single objective and multiobjective meta-heuristic algorithms used in fog environment for addressing various QoS parameters.

Motivation and objective

Literature survey reveals that owing to the NP class nature of the scheduling problem, the work done considering both single-objective and multiobjective problems addressed the scheduling problem in different ways while proposing or utilizing a variety of well-known metaheuristics. However, many of them ignored the need of classifying heterogeneous end-user requests. The present study takes into account the importance of task categorization and is based on the novel approach JS (JellyFish Search Optimiser) [36]. JS has proven to be better than peers e.g. Whale Optimization Algorithm (WOA), Tree-Seed Algorithm (TSA), Symbiotic Organisms Search (SOS), Teaching–Learning-Based Optimization (TLBO), Firefly Algorithm (FA), Gravitation Search Algorithm (GSA), Artificial Bee Colony (ABC), Differential Evolution (DE), Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) algorithms in the mathematical benchmark tests. Additionally, JS has fewer internal parameters resulting in an efficient design, implementation and tuning of the simulation in the proposed work. This work proposes a multi-aspect task scheduler based on the Improved Jellyfish Algorithm (IJFA) which attempts to improve the exploration characteristics of the traditional JS for achieving better scheduling

Table 2 Meta-Heuristics used in fog integrated cloud environment

	Research	Computing model	Optimization technique used	Objective
Single Objective	[47]	Fog	Harris Hawks technique	Energy-saving
	[48]	Fog-Cloud	Genetic Algorithm	Make-span
	[49]	Fog-Cloud	Moth-Flame Optimization	Total task execution time
	[50]	Fog-Cloud	Genetic Algorithm (GA)	Latency
Multiobjective	[51]	Fog	Harris Hawks Technique	Energy Consumption, make-span, cost, flow time, and emission rate of CO_2
	[52]	Fog	Genetic Algorithm	make-span, flow time, fitness function, carbon dioxide emission rate, and energy consumption
	[53]	Fog	Ant colony algorithm	Make-span, response time, and energy usage
	[54]	Fog-Cloud	Particle Swarm Optimization and Cat Swarm Optimization	Execution time, energy consumption, and average response time
	[55]	Fog-Cloud	Genetic Algorithm	Computation time, energy consumption, and more percentage of jobs that meet deadlines
	[56]	Fog-Cloud	Discrete Non-Dominated Sorting Genetic Algorithm II	Makespan and costs
	[57]	Fog-Cloud	Cultural Evolution Algorithm (CEA) and the Invasive Weed Optimization (IWO)	Makespan and energy consumption
	[58]	Fog	Whale Optimization Algorithm	Energy consumption and cost

decisions for the heterogeneous IoT tasks. Further, the classification of jobs in the batch before making scheduling decisions results in a model suitable for real time and deadline aware tasks.

The proposed multi-aspect task scheduling approach

The system's design, problem statement, suggested model and algorithm, simulation setup, and evaluation procedures are all presented in this section. To ensure the effective success of the objectives and aims, the research methodology is built over similar works in the domain of classification of tasks of IoT and its' scheduling employing metaheuristics.

Methodology design

The work aims to schedule the jobs in the fog integrated cloud environment to minimize the make-span time for the jobs. The main research challenge identified is scheduling the requests from various IoT devices with heterogeneous demands in large volumes seeking different QoS needs. The ample search space makes scheduling these requests an even more complicated task. Several previous works and studies have been examined to discover competent qualities that may be utilized to classify and schedule requests. Classification is essential before scheduling for two main reasons: first, the requests are heterogeneous and second, to make use of the integrated environment of cloud and fog layers working with different capabilities and purposes. Fog layer aids real-time processing, necessitating ultra-low latency, location awareness, edge resource pooling, mobility, and most importantly preventing cloud server overloading. It's also helpful in places where connectivity is inconsistent. On the other hand, cloud is more centralized and powerful than fog exhibiting features like availability, computing capabilities and storage capacity. It also aids in the long-term storage of data used for analysis and decision-making. Figure 2 presents the fog integrated cloud system architecture.

All user demands are first submitted to a broker or a gateway with respective parameter values. The centralized broker, which is also a centralized decision-maker, is an intermediate party between the users and the providers and draws its inspiration from the broker used in [59]. The role of the broker is to manage the fog-cloud resource provisioning upon the user requests. It has the same role as a cloud broker, described as "an entity that manages the use, performance and delivery of cloud services, and negotiates relationships between cloud providers and consumers" [60]. The centralized broker records dynamically changing and uncertain resources from multiple fog and cloud providers along with the various IoT devices generating heterogeneous requests. Therefore, it minimizes the complexity arising out of many request

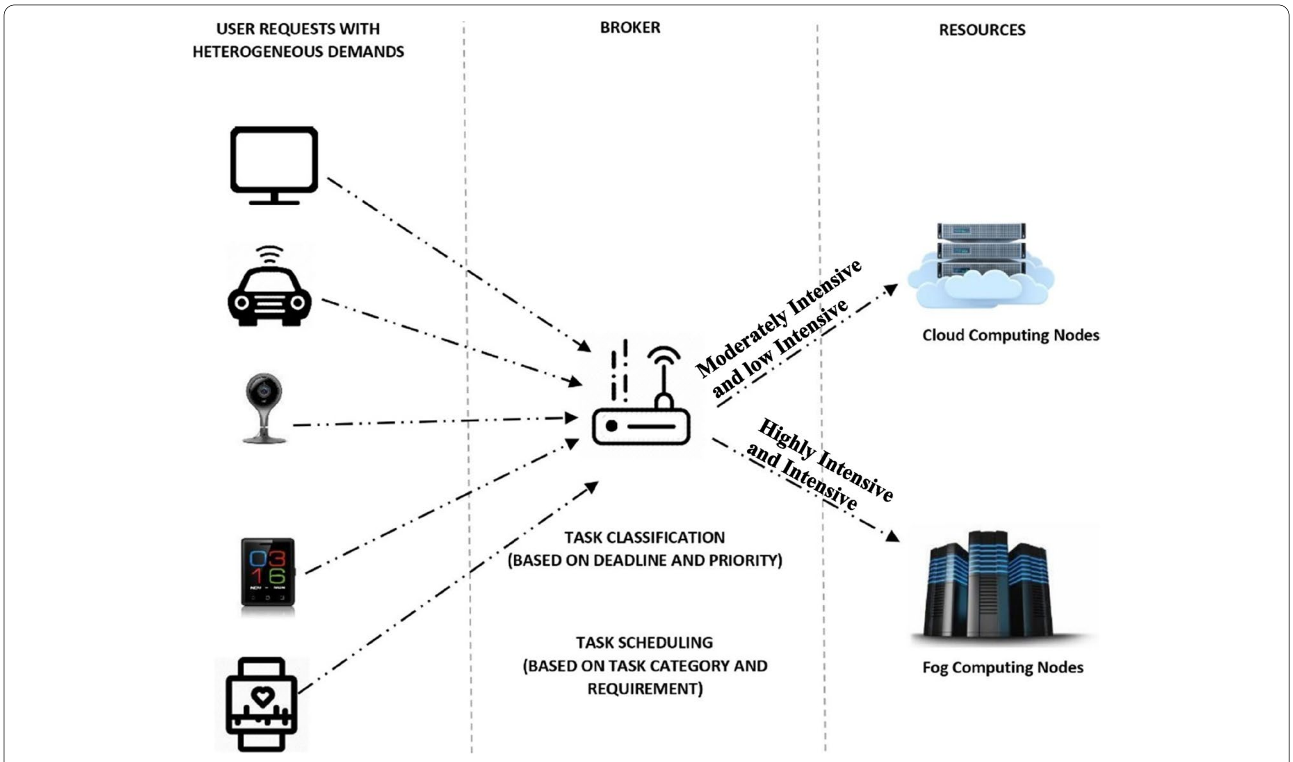
USER REQUESTS WITH HETEROGENEOUS DEMANDS
BROKER
RESOURCES
Moderately Intensive and low Intensive
Cloud Computing Nodes
Highly Intensive and Intensive
Fog Computing Nodes
TASK CLASSIFICATION (BASED ON DEADLINE AND PRIORITY)
TASK SCHEDULING (BASED ON TASK CATEGORY AND REQUIREMENT)

Fig. 2 Fog integrated cloud system architecture

handlers by providing a single point solution in the fog integrated Cloud environment. Once the broker recevices the requests, first, the bi-factor classification algorithm categorizes it based on the deadline and the priority. Next, these tasks are scheduled using the Improved Jellyfish algorithm (IJFA) based on the task category of the requests and their resource requirement to fog nodes or cloud nodes accordingly.

The most important properties of the requests coming from IoT devices are their deadline and priority, which can be used to classify them and then schedule them accordingly. For this purpose, this work proposes a bi-factor classification algorithm to ascertain the task category. The work proposes an Improved Jellyfish Algorithm (IJFA) strengthening the exploration properties of JS algorithm. JS is chosen for improvement as it is a promising meta-heuristic because of its simplicity and the use of only two control parameters, i.e., population size and the number of iterations. Figure 3 presents the workflow of the proposed model.

Problem statement

The IoT devices in the given environment produce data required to be processed for various applications for which it has been deployed. The computationally constrained nature of IoT devices makes it necessary for the devices to offload the data to other computing nodes for processing. In the proposed fog integrated cloud environment, the node assigned with a task and performing computation might be the fog node controlling the network of that geographical location or the cloud node present in a remote location. These nodes consider the processing of data as a task. Each task will be assigned either to a fog node or a cloud node executed as a virtual machine. The VM configuration can be the same if the tasks are of the same type or different depending on the task types. The tasks to for which computation is required can be denoted as,

$$T = \{T_1, T_2, ..T_i \ldots T_k\}; 1, \ldots, i \in k \tag{1}$$

where T_k is the number of tasks submitted by the IoT devices. T_i represents the i^{th} task in the task sequence. These tasks are later clubbed together to form a batch of tasks to be executed as per the QoS requirements. For every task a suitable fog or cloud node is decided for allocation. As with many scheduling approaches, the proposed also aims to minimize the makespan of the tasks submitted by the IoT devices. Each task T_i possesses some features which can be formulated as,

$$T_i = [T_i^{ID}, T_i^{TL}, T_i^{p}, T_i^{dl}] \tag{2}$$

Here,T_i^{ID} is the task identifier, T_i^{TL} the job length (unit: million instructions), T_i^{p} the priority level, and T_i^{dl} deadline of execution of any given task T_i. The nodes which process the tasks can be a fog node ND_f, represented as

$$ND_f = \{N_1, N_2, , , N_j \ldots N_n\}; 1, \ldots, j \in n \tag{3}$$

or a node ND_c in the cloud data center denoted as,

$$ND_c = \{N_1, N_2, , , N_r \ldots N_m\}; 1, \ldots, r \in m \tag{4}$$

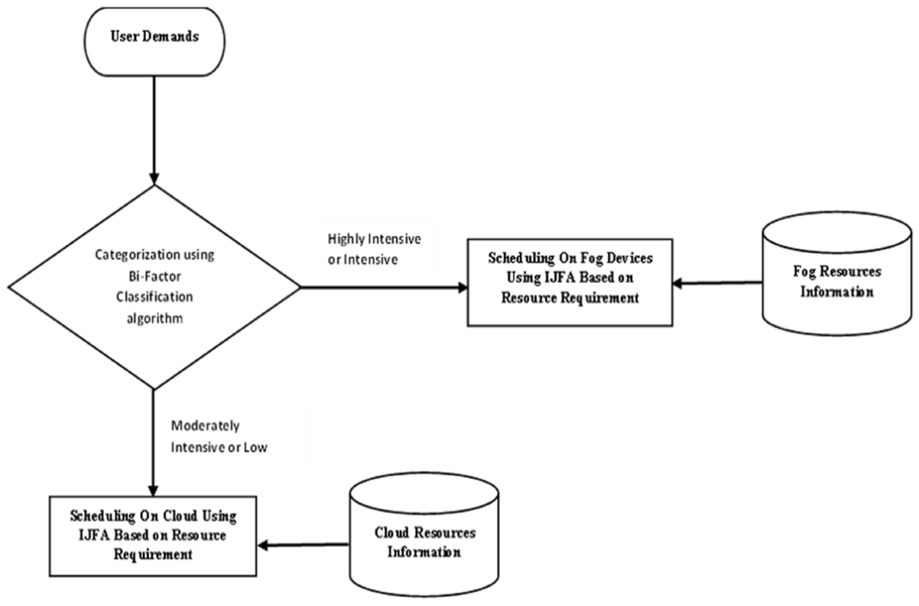
User Demands
Categorization using Bi-Factor Classification algorithm
Highly Intensive or Intensive
Scheduling On Fog Devices Using IJFA Based on Resource Requirement
Fog Resources Information
Moderately Intensive or Low
Scheduling On Cloud Using IJFA Based on Resource Requirement
Cloud Resources Information

Fig. 3 Workflow of the proposed model

Each node N_j and N_r possesses certain features as defined in Eqs. 5 and 6 as

$$N_j = [N_j^{id}, N_j^{MIPS}, N_j^{m}, N_j^{ds}] \tag{5}$$

$$N_r = [N_r^{id}, N_r^{MIPS}, N_r^{m}, N_r^{ds}] \tag{6}$$

Here, N_j^{id} and N_r^{id} denote the identification or the serial number of a node at the fog layer or the cloud layer, N_j^{MIPS} and N_r^{MIPS} the information processing speed of nodes (unit: millions-of-instructions-per-second, MIPS), N_j^{m} and N_r^{m} the memory availability and N_j^{ds} and N_r^{ds} the disk space availability corresponding to nodes N_j and N_r respectively. The allocation of a task T_i to node N_j or N_r obeys certain conditions which are expressed as follows,

$$\sum_{i=1}^{n} T_i^{cpu} \times A_{ij} \le N_j^{cpu} \tag{7}$$

$$\sum_{i=1}^{n} T_i^{m} \times A_{ij} \le N_j^{m} \tag{8}$$

$$\sum_{i=1}^{n} T_i^{ds} \times A_{ij} \le N_j^{ds} \tag{9}$$

Here, A_{ij} is the allocation vector representing the placement of task T_i on node N_j. The above equations describe that the allocation of tasks will be carried out only if the resource availability of the node is greater than the resource requirements of the task. The Expect Complete Time (ECT) matrix of size $[N_k * N_n]$ represents the expected execution time to run the task individually on each computing resource of the fog layer resources as,

$$ECT_f = \begin{matrix} ECT1,1 & ECT1,2 & ECT1,N_n \\ ECT2,1 & ECT2,2 & ECT2,N_n \\ ECTN_k,1 & ECTN_k,2 & ECTN_k,N_n \end{matrix} \tag{10}$$

Similarly, the matrix of size $[N_k * N_m]$ represents the expected execution time to run the tasks on each computing resource (VM) of cloud resources individually as,

$$ECT_C = \begin{matrix} ECT1,1 & ECT1,2 & ECT1,N_m \\ ECT2,1 & ECT2,2 & ECT2,N_m \\ ECTN_k,1 & ECTN_k,2 & ECTN_k,N_m \end{matrix} \tag{11}$$

One of the aims of the proposed model while making a scheduling decision is to minimize the make-span by locating the most efficient group of tasks on virtual machines. $\mathrm{ECT_{ij}}$ for the same can be computed as,

$$ECT_{ij} = \frac{TL(T_i)}{N_j^{MIPS}} i = 1,2,3\ldots.k, j = 1,2,3\ldots.n \tag{12}$$

where ECT_{ij} refers to the required execution time of i th task on j th virtual machine of fog layer, $TL(T_i)$ is the task–length of the i th task. Similarly, ECT_{ir} refers to the expected execution time of i th task on r th VM in the cloud layer represented as Eq. 13.

$$ECT_{ir} = \frac{TL(T_i)}{N_r^{MIPS}} i = 1,2,3\ldots.k, r = 1,2,3\ldots.m \tag{13}$$

The fitness value indicating the suitability of a node for a task can be defined as:

$$fit = max\{ECT_{ij}\}\forall i \in [1,k] \text{mapped to jth } VM, j = 1,2,\ldots n \tag{14}$$

The proposed model aims to optimize the task's execution time and improve the utilization of resources, thereby achieving the required QoS in terms of completion time. The primary objectives for the same can be summarised as,

$$maximize \leftarrow (Ru, QoS) \tag{15}$$

$$minimize \leftarrow (E_T) \tag{16}$$

where, Ru is the resource utilization, QoS the Quality Of Service to be met and E_T the execution time of the tasks/batch of tasks scheduled in the considered computing environment.

Bi-factor task classification

The tasks originating from the IoT devices are aggregated by the $Broker/Gateway$ which classifies tasks and performs scheduling. The incoming tasks possess various priorities and deadlines that should be considered to effectively categorize the tasks. The classification of tasks is performed to analyze the tasks in order to achieve an increased completion rate. The threshold values for priority and deadline of a task can be computed as,

$$H\left(T_i^p\right) = \sum_{i=1}^{n} T_i^p log T_i^p \tag{17}$$

$$H\left(T_i^{dl}\right) = \sum_{i=1}^{n} T_i^{dl} log T_i^{dl} \tag{18}$$

Here, $H\left(T_i^p\right)$ and $H\left(T_i^{dl}\right)$ are the thresholds corresponding to priority and task execution deadline respectively. Further, T_i^p is the priority value and T_i^{dl} is the deadline requirement of the task T_i with the threshold value ranging from 0 to 1. The priority and deadline are classified into two classes, namely low and high, and can be formulated as,

$$H\left(T_i^p\right) = \begin{cases} low, if\, 0 \leq H\left(T_i^p\right) \leq 0.5 \\ high, if\, 0.5 \leq H\left(T_i^p\right) \leq 1 \end{cases} \tag{19}$$

$$H\left(T_i^{dl}\right) = \begin{cases} low, if\, 0 \leq H\left(T_i^{dl}\right) \leq 0.5 \\ high, if\, 0.5 \leq H\left(T_i^{dl}\right) \leq 1 \end{cases} \tag{20}$$

Table 3 presents the task classification based on task priority and the deadline. The classification of tasks is carried out into four major classes: highly intense, intense, moderate, and low, based on the priority and deadline of the tasks. These categorized tasks are then scheduled using the proposed model to meet the desired QoS.

Multi-aspect task scheduling

The scheduling of categorized tasks is performed by the *Broker* to achieve robust computation of tasks. The factors considered for effective scheduling of tasks are task category $\left(T_i^C\right)$, and resource requirement $(T_i^R = T_i^{TL}, T_i^p, T_i^{dl})$.

This work proposes Improved Jellyfish Algorithm (IJFA) to explore the search space for aa better solution for scheduling of tasks with an increased convergence rate. As mentioned earlier, IJFA aims to improve the search space exploration as compared to JS while reducing the convergence time. To computationally realize the same, initially, the Jellyfish vectors are initialized as,

$$\overrightarrow{K}_{\mathfrak{J}+1} = \eta \overrightarrow{K}_{\mathfrak{J}}\left(1 - K_{\mathfrak{J}}\right), 0 \leq \overrightarrow{K}_0 \leq 1 \tag{21}$$

where,$\overrightarrow{K}_0$ denotes the initial vector of the Jellyfish, and $\overrightarrow{K}_{\mathfrak{J}}$ is the vector that holds the chaotic values of $\mathfrak{J}$ th Jellyfish. η is aconstant which is taken as four in the proposed approach. Once the initialization is over, the selection of solution based on Ft is performed to select the location with maximum food $\overrightarrow{K}^*$. Following it, the time control mechanism is utilized to switch between the motions of Jellyfish either towards ocean current or inside the swarms. The motion towards the ocean current can be formulated as,

$$\overrightarrow{K}_{\mathfrak{J}}(t+1) = \overrightarrow{K}_{\mathfrak{J}}(t) + \overrightarrow{rn}.*\left(\overrightarrow{K}^* - \beta * rn_1 * \mu\right) \tag{22}$$

Table 3 Task classification table

Priority	Deadline	Task classified
High	High	Highly Intensive
High	Low	Intensive
Low	High	Moderately intensive
Low	Low	Low intensive

Here, $\overrightarrow{rn}, rn_1$ represents a random number in the range of 0 to 1, β the distribution coefficient and μ is the population mean. The motion of Jellyfish inside the swarm is emulated in two ways using action motion and passive motion. The determination of a new location using passive motion can be computed as,

$$\overrightarrow{K}_{\mathfrak{J}}(t+1) = \overrightarrow{K}_{\mathfrak{J}}(t) + rn_3 * \gamma * \left(up_b - lw_b\right) \tag{23}$$

Here, γ is a constant representing the motion length, rn_3 a random number in the range of 0 to 1, up_b the upper bound of search space and lw_b the lower bound of the search space. The determination of a new location using active motion can be computed as,

$$\overrightarrow{K}_{\mathfrak{J}}(t+1) = \overrightarrow{K}_{\mathfrak{J}}(t) + \overrightarrow{rn} * \overrightarrow{Dr} \tag{24}$$

Here, $\overrightarrow{Dr}$ denotes the motion direction of the Jellyfish to locate the best food and can be expressed as,

$$\overrightarrow{Dr} = \begin{cases} \overrightarrow{K}_{\mathfrak{J}}(t) - \overrightarrow{K}_l(t), ifFt\left(\overrightarrow{K}_{\mathfrak{J}}\right) < Ft\left(\overrightarrow{K}_l\right) \\ \overrightarrow{K}_l(t) - \overrightarrow{K}_{\mathfrak{J}}(t), otherwise \end{cases} \tag{25}$$

In the above equation, l represents the jellyfish index that is selected randomly. The control function $\vartheta(t)$ for switching of motion of Jellyfish can be written as,

$$\vartheta(t) = \left(1 - \frac{t}{t_{max}}\right) * (2 * rn - 1) \tag{26}$$

If the value of $\vartheta(t)$ is greater than or equal to the constant $\vartheta_0,$ the Jellyfish follows the ocean current. Otherwise, the Jellyfish tends to move inside the swarm. If the randomly generated number rn_4 is greater than $1 - \vartheta(t)$ then the Jellyfish moves in a passive motion else it moves in an active motion. Improving JS, the enhancement is in the exploration capabilities of IJFA which prevents premature convergence. IJFA aims to achieving faster convergence while reducing the search time. This can be accomplished by using the randomization concept in investigating the swarm's locations. Initially, the population, as usual is random to have diversity. Later on, it exploits some local optimum, but the introduced randomness also avoids premature convergence. The new formulation can be written as,

$$\begin{aligned} \overrightarrow{K}_{\mathfrak{J}}(t+1) = \overrightarrow{K}_{\mathfrak{J}}(t) + rn * \left(\overrightarrow{K}_{rn_1}(t) - \overrightarrow{K}_{rn_2}(t)\right) \\ + (1 - rn) * \left(K^* - \overrightarrow{K}_{rn_3}(t)\right) \end{aligned} \tag{27}$$

The improved algorithm selects three random solutions from the population and updates each population accordingly to ensure a fast convergence rate. Further, it has a higher chance of moving towards global optimum.

Figure 4 presents the proposed multi-aspect task scheduling using IJFA, which runs at the broker site. The user requests from various IoT devices are submitted and categorized using the bi-classification algorithm and then forwarded to the scheduler. Based on task requirement and category, the scheduler then schedules a task on the fog or cloud layer after initializing the algorithm's parameters and computing and evaluating the fitness of the tasks. The time control mechanism governs the switching between the two types of movement of Jellyfish moving in the ocean in search of food. They are more attracted to locations where the available quantity of food is more significant. The location and corresponding objective function determine the amount of food found.

The pseudocode of the IJFA based task scheduling process is presented in the box below, scheduling of tasks based on task category and resource requirements.

```
Pseudocode

IJFA based task scheduling ()
Input: $T_i^C$ and $T_i^R$
Output: $\vec{K}^*$
Begin
  Perform initialisation
  Perform evaluation of $K_{\mathfrak{J}}$ and select one in $\vec{K}^*$
  Set $t = 1$
  While $t < t_{max}$
    For $\mathfrak{J} = 1{:}N$
      Calculate $\vartheta(t)$ using (43)
      If $\vartheta(t) \geq \vartheta_0$
        Jellyfish move in ocean current using (39)
      Else
        Jellyfish move inside the swarm
        If $rn_4 > \vartheta(t)$
          Perform passive motion using (40)
        Else
          Perform active motion using (41)
        End if
      End if
      Evaluation of $\vec{K}_{\mathfrak{J}}$ and reuse it if found better
      Perform updation of $\vec{K}^*$
      $t = t + 1$
    End for
    For $\mathfrak{J} = 1{:}N$
      Generate $r$ in the range of 0 to 1
      Perform updation of $\vec{K}_{\mathfrak{J}}$ using (44)
    End for
  End while
End
```

Experimental results

This section presents the simulation framework and the experimental result analysis of the proposed model featuring a multi-aspect task scheduler using the Improved Jelly Fish Algorithm (IJFA) for the fog integrated cloud environment. The scheduler uses IJFA to optimize the scheduling of IoT tasks based on the task's priority, deadline and resource requirements. The requests are classified based on deadline and priority and then accordingly scheduled on either the fog or the cloud layer.

Simulation environment

Experimenting on real-world data in a real-world cloud environment is costly and complicated. Furthermore, if the experiment does not go as intended, crucial data may be lost. As a result, a simulator that can simulate the real-world cloud environment is necessary to test and validate various strategies and procedures presented for the benefit of the stakeholders. Once approved, these strategies and mechanisms can be used in a real-world context without the risk of data loss.

The classification before scheduling across the data center is essential in IoT tasks due to its effectiveness in managing the execution by executing real-time requests with high priority within their deadline. This work uses a Bi-factor classification algorithm to classify requests before scheduling them to address the concern. The experimental parameters are based on previous literature work reported in the literature for cloud and fog based resource provisioning.

The experiments were conducted by simulating a hybrid fog-cloud environment and heterogeneous task transfer. Table 4 summarises the simulation attributes used in the work. For effective simulation, MATLAB toolbox named Distributed System is utilized. This toolbox can be used to emulate any distributed network including fog integrated cloud environment. All experiments were executed for twenty different runs, including 3000 iterations for optimization.

The different batch sizes of Real-Time (RT) and Non-Real Time (NRT) requests were randomly generated and submitted. The distribution of VMs for fog and cloud nodes was kept at 30% and 70%, respectively. In each simulation, VMs, cloud and fog nodes were randomly generated with different processing capacities and configurations within the range in accordance with Table 5 as presented in the coming section.

Experimental setting

The proposed work used MATLAB to realize the fog integrated cloud environment. The experiments were designed to use five batches comprising of 600,

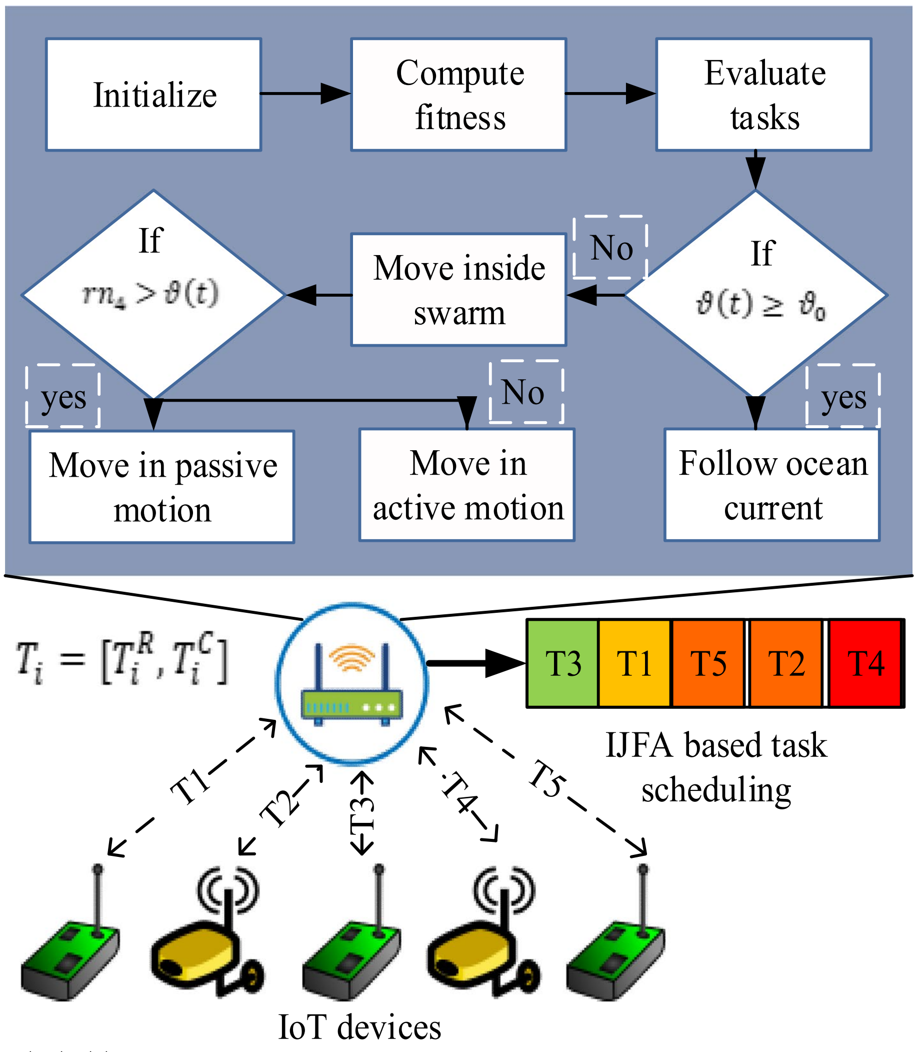

Fig. 4 Multi-aspect task scheduling using IJFA

1200,1800, 2400 and 3000 tasks respectively. The different dataset sizes allow for more complicated task transfer scheduling testing. As the dataset size grows larger, the scheduling difficulty becomes more difficult, thus ensuring rigorous testing of the model under a real fog-cloud like workload. Each task is represented by four parameters: task ID, task size, task deadline, and the task's priority. Task ID is a unique transfer identifier while job size provides an estimate of the computational execution time. Expected Completion Time (ECT) is used as the parameter to compare the performance of both JS and IJFA. In the computing environment, the dataset placement has been evaluated for three different scenarios: (1) allocating the dataset for execution directly on the cloud, (2) allocating the dataset for execution directly on the fog, (3) allocating the dataset for execution across fog

Table 4 Simulation attributes

Entity	Parameter	Value of settings
Requests/Tasks	Number of tasks	600–3000
	Length	10–50 (Mb)
	Priority	High/Low
	Deadline	High/Low
Virtual machine (Cloud)	Number of VMs	30% of the total on cloud
	Processing Capacity range	[100,1000] (MIPS)
Virtual machine (Fog)	Number of VMs	70% of the total on Fog
	Processing Capacity range	[10,100] (MIPS)

Table 5 Summary of ECT observations for varying number of tasks for JS and IJFA

S.no	Tasks		JS		IJFA	
			Best	**Worst**	**Best**	**Worst**
1	600	Trial 1	19.1567	21.5287	14.6579	21.5287
		Trial 2	23.0880	25.1006	16.3882	25.1006
		Trial 3	10.3659	17.0845	10.5985	16.6051
		Average	**17.5369**	**21.2379**	**13.8815**	**21.0781**
2	1200	Trial 1	40.8966	43.0645	36.0634	43.0645
		Trial 2	20.3129	33.8435	20.5149	33.9805
		Trial 3	30.5456	44.9759	30.3910	44.0280
		Average	**30.5850**	**40.6280**	**28.9898**	**40.3577**
3	1800	Trial 1	58.6918	63.9129	52.7093	63.9575
		Trial 2	32.6999	45.9925	29.2666	49.0941
		Trial 3	40.8887	74.6994	38.3106	74.6994
		Average	**44.0935**	**61.5349**	**40.0955**	**62.5837**
4	2400	Trial 1	46.4022	66.4697	48.3129	65.7420
		Trial 2	60.3213	101.4393	66.074	101.4393
		Trial 3	81.608	88.1454	61.0000	87.6158
		Average	**62.7772**	**85.3515**	**58.4623**	**84.9324**
5	3000	Trial 1	86.3706	95.3073	75.8116	96.2316
		Trial 2	92.8842	131.8479	94.0386	132.5835
		Trial 3	115.8123	120.5761	100.0861	120.5761
		Average	**98.3557**	**115.9104**	**89.9788**	**116.4637**

integrated cloud architecture using classification being the main focus of this work.

Experimental results and analysis

Experiments using meta-heuristic scheduling algorithms, JS and the proposed IJFA were conducted in the fog-cloud environment. The findings of the experimentation are presented in this section. The simulation's goal is to calculate the make-span of these two meta-heuristic scheduling methods being tested. The ideal meta-heuristic scheduling method is believed to be the one with the shortest make-span. Different test scenarios are used to calculate the performance of these two based on the ECT matrix to ensure the VMs' balanced workload. In the proposed model, the task scheduling phase is proceeded by the bi-factor classification phase, which improves the overall performance by characterizing user tasks based on the priority and the deadline. Accordingly, the incoming tasks in the batch of jobs are classified into highly intensive, intensive, moderately intensive, and less intensive based on the priority and deadline. These two parameters enable the deadline sensitive and essential tasks to be scheduled at the fog layer, decreasing latency and time delay. These categorized tasks are then considered for scheduling, aiming to achieve better makespan.

Scenario 1: batch size variation from 600 to 3000 tasks with a fixed number of iterations as 3000

In this case, five different batches of tasks comprising tasks varying in the range 600–3000 were scheduled using JS and IJFA executed over 100 VMs run over 3000 iterations. Figure 5a-e presents the Expected Completion Time (ECT) of an optimized schedule for all the five batches of tasks having 600, 1200, 1800, 2400 and 3000 tasks, respectively based on JS and IJFA algorithms through the convergence curves. As discussed in The proposed multi-aspect task scheduling approach section, the tasks are classified into four categories, i.e., highly intensive, intensive, moderately intensive and low intensive, based on deadline and priority using a bi-classification algorithm. These are then offloaded to the cloud and fog layer of the hybrid architecture based on their category. In the proposed model, on average, almost half of the requests generated in different trials are getting served by the fog, which directly improves the system's total performance. The scheduler then, using JS and IJFA, allocates requests on available fog nodes by calculating the expected time of completion of each request of the batch and selects the one with minimum time among the available nodes at the respective layer where they have been offloaded to ensure a balanced workload for VMs. An average ECT scorewas computed for virtual

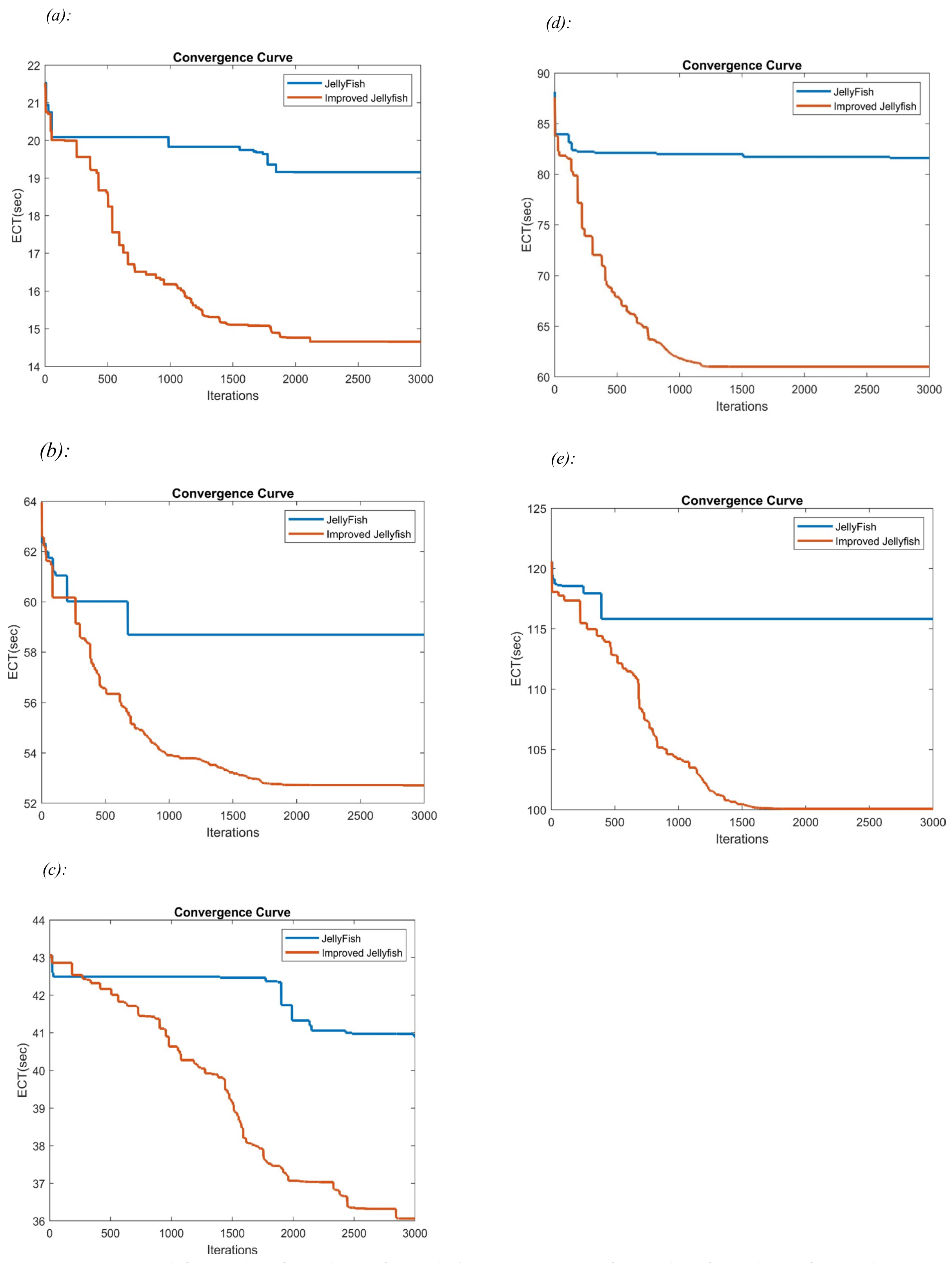

Fig. 5 **a** Convergence Graph for JS and IJFA for Batch Size of 600 Tasks. **b** Convergence Graph for JS and IJFA for Batch Size of 1200 Tasks. **c** Convergence Graph for JS and IJFA for Batch Size of 1800 Tasks. **d** Convergence Graph for JS and IJFA for Batch Size of 2400 Tasks. **e** Convergence Graph for JS and IJFA for Batch Size of 3000 Tasks

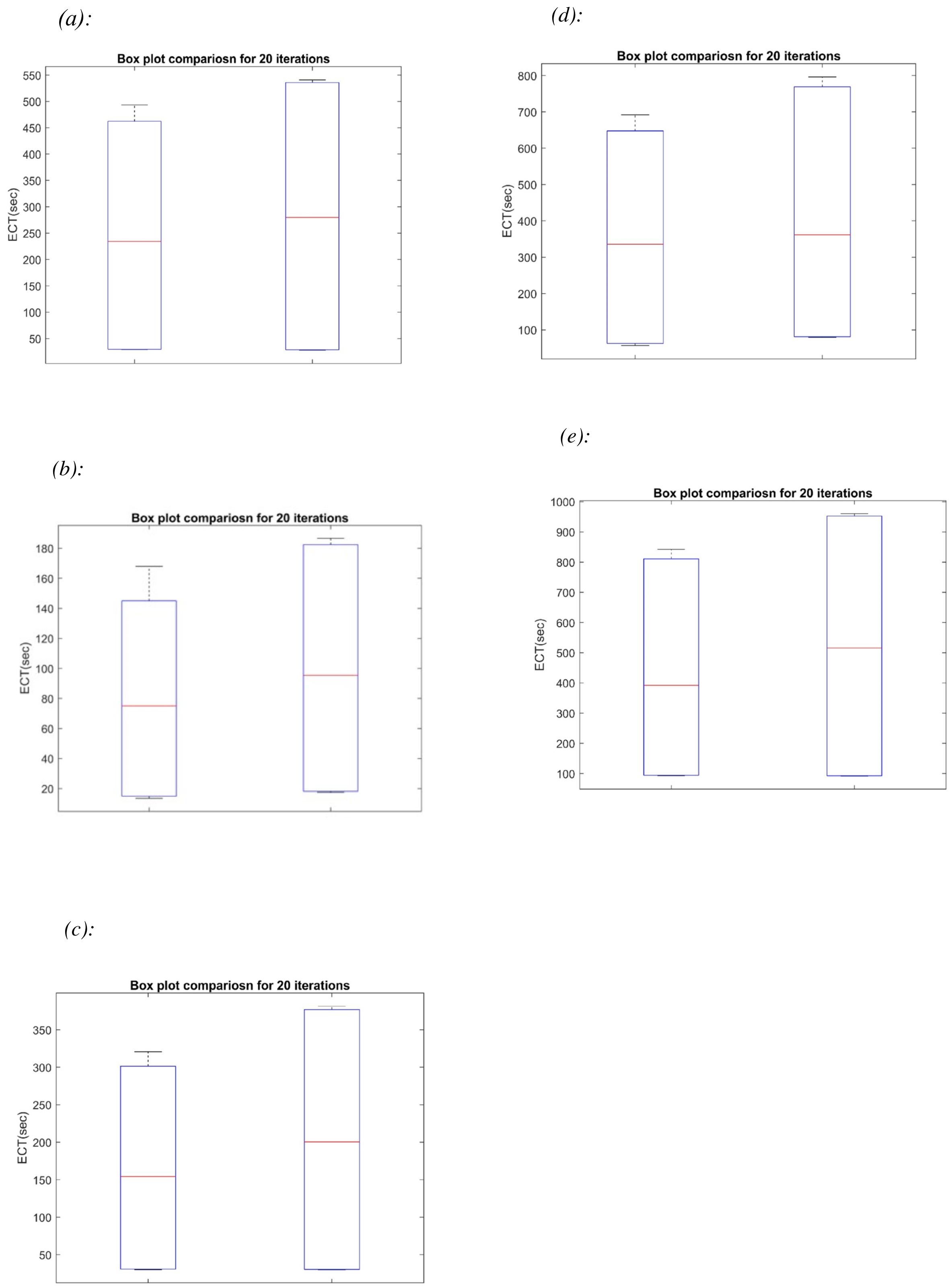

Fig. 6 **a** Comparison of IJFA and JS for 20 iterations for 600 Tasks. **b** Comparison of IJFA and JS for 20 iterations for 1200 Tasks. **c** Comparison of IJFA and JS for 20 iterations for 1800 Tasks. **d** Comparison of IJFA and JS for 20 iterations for 2400 Tasks. **e** Comparison of IJFA and JS for 20 iterations for 3000 Tasks

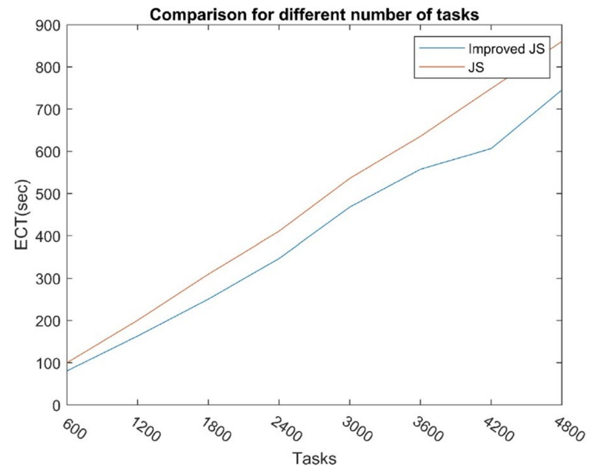

Fig. 7 ECT v/s number of tasks

machines at cloud and fog layers using the JS algorithm. From Fig. 5a-e, it is observed that The IJFA can converge to a better solution with a better convergence rate in all the cases, from smaller batches of tasks to bigger ones. The better convergence of IJFA can be attributed to its better exploration capability to achieve faster convergence by reducing the search time than the traditional JS. This was made possible in IJFA by exploring search locations or solutions which other members could not have previously explored by randomly picking any three solutions from the population and updating the population accordingly.

Table 5 summarises the optimized ECT based on the various experiment scenarios. The results were obtained for every set of tasks for three different runs. The average best and average worst performance in ECT reported by JS and IJFA were observed. The results based on these three random trials over different task sets indicate that the optimized ECT observed in the model using the proposed IJFA is way lesser than the one obtained with JS as a trend due to efficient task scheduling based on multiple aspects by IJFA including bi-classification. The IJFA has faster convergence due to the improvement in the exploration phase and achieves the trade-off between exploration and exploitation with less time consumption due to implementing a two-directional search strategy resulting in better solutions. For instance, in the experiment with 600 batches of tasks, the optimized ECT for IJFA is about 13 s in the average best scenario, which is better than the average best of 17 s reported by JS. As far as the average worst scenario is concerned for both JS and IJFA, it is observed that IJFA is performing at par with JS. The same trend can be seen in all the different batches of jobs. It can be concluded that IJFA outperforms JS easily when used in the fog integrated cloud environment with randomly generated and scheduled heterogeneous tasks.

The ECT of JS and IJFA implementations for various task sets spread over 20 iterations is presented in Fig. 6a–e. A boxplot performance for all five batches of jobs has been given. It is observed that the median of IJFA is less than JS in all the cases. For instance, the median for batch I (600 tasks 100 VMs) is 75 s for IJFA and 95 s for traditional JS. Also, the ECT for IJFA and JS has been dispersed over 167 s to 13 s and 186 s to 17 s, respectively. The same trend has been observed in all the other four batches of jobs, too, where range dispersion and median of IJFA are less than JS. The box plots reflects that IJFA performs an efficient task scheduling compared to JS, irrespective of the task set sizes. The result remains the same even with the change in the number of VMs. The same has not been reproduced here to avoid redundancy.

The performance of the proposed IJFA algorithm and traditional JS has also been evaluated by feeding a different number of tasks ranging from 600 to 4800to observe a general trend of the performance of the two algorithms. The same has been presented in Fig. 7. This is significant because a wide range of data sizes could be used to support the task transfer scheduling's sophisticated testing. As the data size grows larger, the scheduling becomes more difficult but still IJFA performes very well as compared to traditional JS. It is worth mentioning that the performance of IJFA becomes even superior with the increasing task set size which corresponding with the fog integrated cloud environment. As the size of the batch increases, the difference in ECT reported by both

Table 6 ECT v/s Batch Size for Fog Integrated Cloud, Cloud-Only and Fog-Only Architectures

Task size	Fog integrated cloud		Cloud-only		Fog-only	
	JS	IJS	JS	IJS	JS	IJS
Batch I	17.5369	13.8815	37.08718	31.89165	402.4644	321.3767
Batch II	30.58503	28.98977	74.02684	64.11581	781.0749	678.7809
Batch III	44.0935	40.0955	103.9711	100.7851	973.1735	936.5133
Batch IV	62.77717	58.4623	146.9762	127.7616	1541.985	1475.785
Batch V	98.3557	89.97877	183.0033	161.995	1796.823	1664.369

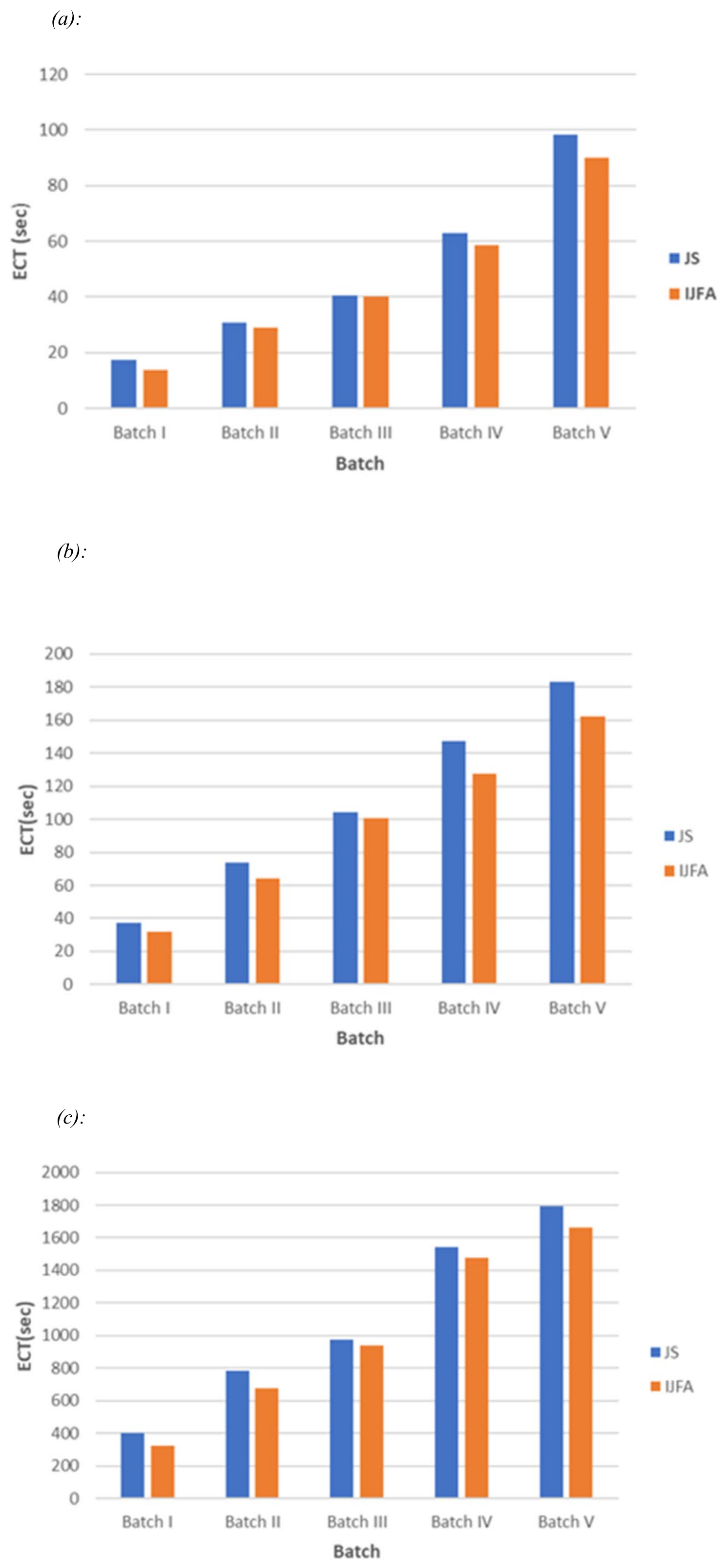

Fig. 8 **a** ECT v/s Batch Size for JS and IJFA for the Fog Integrated Cloud Architecture. **b** ECT v/s Batch Size for JS and IJFA for the Cloud Only Architecture. **c** ECT v/s Batch Size for JS and IJFA for the Fog-Only Architecture

Table 7 ECT v/s batch size for IJFA for fog integrated cloud, cloud-only and fog-only architectures

Task size	Fog integrated cloud	Cloud-only	Fog-only
Batch I	13.8815	31.89165	321.3767
Batch II	28.98977	64.11581	678.7809
Batch III	40.0955	100.7851	936.5133
Batch IV	58.4623	127.7616	1475.785
Batch V	89.97877	161.995	1664.369

the algorithms also increases. The increased difference between the two concluded that IJFA is more suitable in the situation with large size datasets, which is very appropriate in the real-world scenarios especially considering IoT devices as the task generators.

Scenario 2: performance evaluation of IJFA using Fog Integrated Cloud, Cloud-Only and Fog- Only Scheduling architectures

To understand the usefulness of fog- integrated cloud architecture over cloud-only and fog-only architectures, JS and the proposed algorithm IJFA was applied to work in three scenarios for the above mentioned three architectures. To realize the same, five batches of tasks with batch sizes ranging from 600 to 3000 were made, which were later scheduled on the above three architectures separately. Accordingly, batches were made with batch I comprising 600 tasks, batch II comprising 1200 tasks till batch V comprising 3000 tasks. A comparative analysis of the minimized makes-span trend for this study has been presented in Table 6 and a bar chart representation is depicted in Fig. 8a-c. The ECT observed in both the cases of using JS and IJFA clearly outlines the outperformance of IJFA over JS for all the batch sizes over randomly generated datasets.

Table 7 summarises the results obtained using IJFA as the scheduling strategy with the results in terms of ECT for varying batch sizes for the fog-only, cloud-only and the fog integrated cloud architectures. These results have been presented pictorially in Fig. 9. It is observed that the low-capacity nodes at the fog layer led to maximum ECT in all the used five batches of jobs followed by cloud-only architecture. For instance, ECT for the batch I dataset came out to be approximately 13 s, 31 s and 321 s for fog integrated cloud, cloud-only and fog-only architectures The same trend can be seen for the remaining batches too. Therefore, it is established that the proposed IJFA algorithm performs very well in the hybrid fog-cloud environment owing to the classification of jobs prior to actual scheduling, which led to offloading of high priority and time-sensitive requests to the fog nodes and rest to the cloud nodes. As per the analysis of the random generation of requests in different trials, almost 50% of jobs were executed at the fog layer, resulting in a low ECT in the hybrid environment. In all the cases, if all the tasks were offloaded to the fog layer, it resulted in the maximum ECT as the number of fog devices is limited and computationally constrained as compared to cloud-only and fog integrated cloud architecture. The study infers that it is best to have an integration of fog nodes and cloud nodes contributing to schedule based on the task preferences to achieve better performance.

This section studies the experimental results of the proposed model to optimize the execution of IoT tasks in the fog integrated cloud architecture. The effect of the batches of data on the performance was analyzed to study the results under various scenarios. It is gathered from the experimental results that the optimization

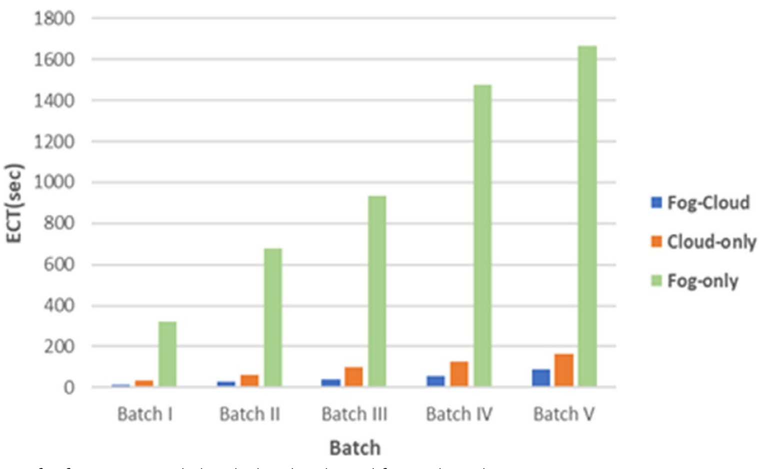

Fig. 9 ECT v/s batch size for IJFA for fog integrated cloud, cloud-only and fog-only architectures

of tasks in fog-only and cloud-only architecture faced challenges in scheduling as they did not classify and categorized the tasks based on their requirements and nature. This led to an increased ECT in both the scenarios. Also, with an increased number of tasks, the complexity of job scheduling increases, leading to delays in response and in the worst case request failures. The integration of classification of jobs with IJFA results in an efficient ECT reduction of the batch of jobs under numerous testing cases. Compared to using solely standard JS algorithm, the proposed model comprising of two-phase scheduling algorithm allows for adequate optimization time. The bi-classification phase helps in characterizing the IoT requests based on the deadline and the priority to schedule the tasks using IJFA on the available computing resources. These available resources can be either at the cloud layer or fog layer with their selection ensuring an optimal use of both for an efficient job execution. Therefore, this work gains significance as it addresses the efficient resource provisioning in the fog-ingrated cloud environment for the heterogeneous and delay sensitive IoT tasks to reduce the execution time leading to a decreased cost of execution for the batch of jobs.

Conclusion and future works

The Internet of Things (IoT) has emerged as a prominent technology in automation by performing real-time tasks for various applications with different priorities and deadlines. An integrated fog-cloud architecture offers the remote execution of these tasks with an improved latency and better availability of bandwidth. However, it is important to have a suitable resource provisioning scheme for task offloading to fog and cloud resources while meeting the task and resource requirements. This work proposes a multi-aspect task scheduling algorithm based on Improved Jellyfish Algorithm (IJFA) by leveraging the benefits of the integrated environment of fog-cloud. The proposed bi-factor classification phase helps the tasks to be scheduled to the right place based on their deadline and priority. The scheduler ensures minimizing the task completion time while maximizing resource utilization thus improving the overall QoS. The simulation results reveal that combining the classification phase with the IJFA speed up the completion of tasks based on their relevance. The advantage of using fog resources integrated with cloud resources offering superior task completion time has been presented. The model was rigorously tested to successfully evaluate its performance from low load to the real-world scenario of large volume and heterogeneous IoT task requests. The proposed IJFA based scheduler outperforms the traditional Jellyfish search optimizer (JS) under various test conditions to prove its effectiveness.

The authors have already started to take this work further to propose a multiobjective model minimizing the response time and energy for the integrated fog-cloud architecture aiming an improved QoS to the users while decreasing the cost for the providers.

Acknowledgements
Not applicable.

Authors' contributions
Nupur Jangu conceived of the presented idea and developed the theory and performed the computations. Zahid Raza verified the analytical methods and supervised the findings of this work. All authors discussed the results and contributed to the preparation of the final manuscript. The author(s) read and approved the final manuscript.

Authors' information
Ms. Nupur Jangu, School of Computer and Systems Sciences, Jawaharlal Nehru University, India (jangu.nupur21@gmail.com)
Lab 02
School of Computer and Systems Sciences
New Delhi, Delhi 110,067, IN
Nupur Jangu is pursuing PhD in the School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi, India. She received her MCA (Master of Computer Applications) and BCA (Bachelor of Computer Applications) degree from Rajasthan Technical University, Rajasthan in the year 2014 and 2011, respectively. She has a teaching experience of about three years. Her areas of interests are cloud computing, Internet of Things (IoT), fog computing, optimization and parallel & distributed computing
Dr. Zahid Raza, School of Computer and Systems Sciences, Jawaharlal Nehru University, India (zahidraza75@gmail.com)
Lab 02
School of Computer and Systems SciencesNew Delhi, Delhi 110,067, IN
Dr. Zahid Raza is currently serving as a Professor in the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. Dr. Raza has a M.Sc degree in Electronics in which he was the Gold Medalist, M.Tech. degree in Computer Science and Ph.D. in Computer Science. Prior to joining Jawaharlal Nehru University, he served as a Lecturer in Banasthali Vidyapith University, Rajasthan, India. His research interest is in the area of Parallel and Distributed Systems. He has proposed various scheduling models for job scheduling for Computational Grid, Cloud and Parallel Systems. His research interest also includes the use of machine learning for medical problems.
Dr. Raza has published many research papers in various peer reviewed International Journals and Transactions. He has various publications in proceedings of various peer-reviewed conferences in India and abroad. Dr. Raza is one of the authors of the Springer Briefs in Computer Science entitled *Auction based Resource Provisioning in Cloud Computing*. He has also contributed a chapter in an edited book. Various invited talks have been delivered by him throughout his academic career. Dr. Raza has also served in various committees for various academic, administrative and evaluation purposes

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 9 May 2022 Accepted: 3 December 2022
Published online: 21 December 2022

References

1. Afshar A, Massoumi F, Afshar A, Mariño MA (2015) State of the Art Review of Ant Colony Optimization Applications in Water Resource Management. Water Resour Manage 29:3891–3904. https://doi.org/10.1007/s11269-015-1016-9
2. Banks A, Vincent J, Anyakoha C (2007) A review of particle swarm optimization. Part I: background and development. Nat Comput 6:467–484. https://doi.org/10.1007/s11047-007-9049-5
3. Fister I, Yang XS, Brest J (2013) A comprehensive review of firefly algorithms. Swarm Evol Comput 13:34–46. https://doi.org/10.1016/J.SWEVO.2013.06.001
4. Fister I, Perc M, Kamal SM, Fister I (2015) A review of chaos-based firefly algorithms: Perspectives and research challenges. Appl Math Comput 252:155–165. https://doi.org/10.1016/J.AMC.2014.12.006
5. Yang XS (2014) Preface. Studies in Computational. Intelligence 585:v–vi. https://doi.org/10.1007/978-3-319-02141-6
6. Hussain K, Mohd Salleh MN, Cheng S, Shi Y (2019) Metaheuristic research: a comprehensive survey. Artif Intell Rev 52:2191–2233. https://doi.org/10.1007/s10462-017-9605-z
7. Yang X-S, Chien SF, Ting TO (2014) Computational Intelligence and Metaheuristic Algorithms with Applications. Sci World J 2014:425853. https://doi.org/10.1155/2014/425853
8. Fister I, Yang XS, Brest J, Fister D (2013) A brief review of nature-inspired algorithms for optimization. Elektroteh Vestn/Electrotech Rev 80:116–122
9. Soltanshahi M, Asemi R, Shafiei N (2019) Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers. Heliyon 5:e02066. https://doi.org/10.1016/J.HELIYON.2019.E02066
10. Kesavaraja D, Shenbagavalli A (2018) QoE enhancement in cloud virtual machine allocation using Eagle strategy of hybrid krill herd optimization. J Parallel Distrib Comput 118:267–279. https://doi.org/10.1016/J.JPDC.2017.08.015
11. Usman MJ, Ismail AS, Chizari H et al (2019) Energy-efficient Virtual Machine Allocation Technique Using Flower Pollination Algorithm in Cloud Datacenter: A Panacea to Green Computing. J Bionic Eng 16:354–366. https://doi.org/10.1007/s42235-019-0030-7
12. Liu XF, Zhan ZH, Deng JD et al (2018) An Energy Efficient Ant Colony System for Virtual Machine Placement in Cloud Computing. IEEE Trans Evol Comput 22:113–128. https://doi.org/10.1109/TEVC.2016.2623803
13. Alresheedi SS, Lu S, Abd Elaziz M, Ewees AA (2019) Improved multiobjective salp swarm optimization for virtual machine placement in cloud computing. HCIS 9:15. https://doi.org/10.1186/s13673-019-0174-9
14. Li G, Wu Z Ant Colony Optimization Task Scheduling Algorithm for SWIM Based on Load Balancing. https://doi.org/10.3390/fi11040090
15. Natesan G, Chokkalingam A (2019) Optimal task scheduling in the cloud environment using a mean Grey Wolf Optimization algorithm. Int J Tech 10:126–136. https://doi.org/10.14716/ijtech.v10i1.1972
16. Sreenu K, Sreelatha M (2019) W-Scheduler: whale optimization for task scheduling in cloud computing. Cluster Comput 22:1087–1098. https://doi.org/10.1007/s10586-017-1055-5
17. Huang X, Li C, Chen H, An D (2020) Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies. Cluster Comput 23:1137–1147. https://doi.org/10.1007/s10586-019-02983-5
18. Chaudhary D, Singh Chhillar R (2013) A New Load Balancing Technique for Virtual Machine Cloud Computing Environment. Int J Comput Appl 69:37–40. https://doi.org/10.5120/12114-8498
19 Mohammad OKJ (2018) GALO: A new intelligent task scheduling algorithm in cloud computing environment. Int J Eng Technol (UAE) 7:2088–2094. https://doi.org/10.14419/ijet.v7i4.16486
20. Chaudhary D, Kumar B (2018) Cloudy GSA for load scheduling in cloud computing. Appl Soft Comput 71:861–871. https://doi.org/10.1016/J.ASOC.2018.07.046
21. Kaur M, Kadam S (2018) A novel multiobjective bacteria foraging optimization algorithm (MOBFOA) for multiobjective scheduling. Appl Soft Comput 66:183–195. https://doi.org/10.1016/J.ASOC.2018.02.011
22. Elaziz MA, Xiong S, Jayasena KPN, Li L (2019) Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowl Based Syst 169:39–52. https://doi.org/10.1016/J.KNOSYS.2019.01.023
23. Rajagopalan A, Modale DR, Senthilkumar R (2020) Optimal Scheduling of Tasks in Cloud Computing Using Hybrid Firefly-Genetic Algorithm. In: Satapathy SC, Raju KS, Shyamala K et al (eds) Advances in Decision Sciences, Image Processing, Security and Computer Vision. Springer International Publishing, Cham, pp 678–687
24. Pradeep K, Prem Jacob T (2018) A Hybrid Approach for Task Scheduling Using the Cuckoo and Harmony Search in Cloud Computing Environment. Wireless Pers Commun 101:2287–2311. https://doi.org/10.1007/s11277-018-5816-0
25. Gabi D, Samad Ismail A, Zainal A, et al Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. https://doi.org/10.1007/s00521-016-2816-4
26. Gobalakrishnan N, Arun C (2018) A New Multi-Objective Optimal Programming Model for Task Scheduling using Genetic Gray Wolf Optimization in Cloud Computing. Comput J 61:1523–1536. https://doi.org/10.1093/comjnl/bxy009
27. Abualigah L, Alkhrabsheh M (2022) Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. J Supercomput 78:740–65. https://doi.org/10.1007/s11227-021-03915-0
28. Jeddi S, Sharifian S A water cycle optimized wavelet neural network algorithm for demand prediction in cloud computing. https://doi.org/10.1007/s10586-019-02916-2
29. Jayasena KPN, Li L, AbdElaziz M, Xiong S (2018) Multi-objective Energy Efficient Resource Allocation Using Virus Colony Search (VCS) Algorithm. 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). pp 766–773
30. Hamid Hussain Madni S, Shafie Abd Latiff M, Abdulhamid M, Ali J Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment. https://doi.org/10.1007/s10586-018-2856-x
31. Elsherbiny S, Eldaydamony E, Alrahmawy M, Reyad AE (2018) An extended Intelligent Water Drops algorithm for workflow scheduling in cloud computing environment. Egypt Inform J 19:33–55. https://doi.org/10.1016/j.eij.2017.07.001
32. Manasrah AM, Ba Ali H (2018) Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing. Wirel Commun Mob Comput 2018:1934784. https://doi.org/10.1155/2018/1934784
33. Karthikeyan K, Sunder R, Shankar K et al (2020) Energy consumption analysis of Virtual Machine migration in cloud using hybrid swarm optimization (ABC-BA). J Supercomput 76:3374–3390. https://doi.org/10.1007/s11227-018-2583-3
34. Kalra M, Singh S (2015) A review of metaheuristic scheduling techniques in cloud computing. Egypt Inform J 16:275–295. https://doi.org/10.1016/J.EIJ.2015.07.001
35. Consortium O, Working A (2017) Open fog reference architecture for fog computing. Open Fog Consortium Architecture Working Group. pp 1–162
36. Chou JS, Truong DN (2021) A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. Appl Math Comput 389:125535. https://doi.org/10.1016/j.amc.2020.125535
37. Houssein EH, Gad AG, Wazery YM, Suganthan PN (2021) Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends. Swarm Evol Comput 62:100841. https://doi.org/10.1016/J.SWEVO.2021.100841
38. Mandal T, Acharyya S (2015) Optimal task scheduling in cloud computing environment: Meta heuristic approaches. 2015 2nd International Conference on Electrical Information and Communication Technologies (EICT). pp 24–28
39. Raju R, Babukarthik RG, Chandramohan D et al (2013) Minimizing the makespan using Hybrid algorithm for cloud computing. 2013 3rd IEEE International Advance Computing Conference (IACC). pp 957–962
40. Zuo L, Shu L, Dong S, et al Special section on big data services and computational intelligence for industrial systems A Multiobjective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing. https://doi.org/10.1109/ACCESS.2015.2508940
41. Ramezani F, Jie, Farookh L et al (2014) Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization. Int J Parallel Prog 42:739–754. https://doi.org/10.1007/s10766-013-0275-4
42. He H, Xu G, Pang S, Zhao Z (2016) AMTS: Adaptive multiobjective task scheduling strategy in cloud computing. China Commun 13:162–171. https://doi.org/10.1109/CC.2016.7464133

43. Chaudhary D, Kumar B, Khanna R (2017) NPSO Based Cost Optimization for Load Scheduling in Cloud Computing. In: Thampi S, Martínez Pérez G, Westphall C, Hu J, Fan C, Gómez Mármol F. (eds) Security in Computing and Communications. SSCC 2017. Communications in Computer and Information Science, vol 746. Springer, Singapore. https://doi.org/10.1007/978-981-10-6898-0_9
44. Ramezani F, Lu J, Taheri J et al (2015) Evolutionary algorithm-based multiobjective task scheduling optimization model in cloud environments. World Wide Web 18:1737–1757. https://doi.org/10.1007/s11280-015-0335-3
45. Hamid Hussain Madni S, Shafie Abd Latiff M, Ali J, Abdulhamid M (2019) Multi-objective-Oriented Cuckoo Search Optimization-Based Resource Scheduling Algorithm for Clouds. Arab J Sci Eng 44:3585–3602. https://doi.org/10.1007/s13369-018-3602-7
46. Wu Z, Liu X, Ni Z et al (2013) A market-oriented hierarchical scheduling strategy in cloud workflow systems. J Supercomput 63:256–293. https://doi.org/10.1007/s11227-011-0578-4
47. AL-Amodi S, Patra SS, Bhattacharya S, Mohanty, JR, Kumar V, Barik RK (2022) Meta-heuristic Algorithm for Energy-Efficient Task Scheduling in Fog Computing. In: Dhawan A, Tripathi VS, Arya KV, Naik K. (eds) Recent Trends in Electronics and Communication. Lecture Notes in Electrical Engineering, vol 777. Springer, Singapore. https://doi.org/10.1007/978-981-16-2761-3_80
48. Liu Q, Wei Y, Leng S, Chen Y (2017) Task scheduling in fog enabled Internet of Things for smart cities. 2017 IEEE 17th International Conference on Communication Technology (ICCT). pp 975–980
49. Ghobaei-Arani M, Souri A, Safara F, Norouzi M (2020) An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing. Trans Emerg Telecommun Technol 31:1–14. https://doi.org/10.1002/ett.3770
50. Aburukba RO, AliKarrar M, Landolsi T, El-Fakih K (2020) Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing. Future Gener Comput Syst 111:539–551. https://doi.org/10.1016/j.future.2019.09.039
51. Abdel-Basset M, El-Shahat D, Elhoseny M, Song H (2021) Energy-Aware Metaheuristic Algorithm for Industrial-Internet-of-Things Task Scheduling Problems in Fog Computing Applications. IEEE Internet Things J 8:12638–12649. https://doi.org/10.1109/JIOT.2020.3012617
52. Abdel-Basset M, Mohamed R, Chakrabortty RK, Ryan MJ (2021) IEGA: An improved elitism-based genetic algorithm for task scheduling problem in fog computing. Int J Intell Syst 36:4592–4631. https://doi.org/10.1002/int.22470
53. Ghaffari E (2019) Providing a new scheduling method in fog network using the ant colony algorithm
54. Rafique H, Shah MA, Islam SU et al (2019) A Novel Bio-Inspired Hybrid Algorithm (NBIHA) for Efficient Resource Management in Fog Computing. IEEE Access 7:115760–115773. https://doi.org/10.1109/ACCESS.2019.2924958
55. Hoseiny F, Azizi S, Shojafar M et al (2021) PGA: A Priority-aware Genetic Algorithm for Task Scheduling in Heterogeneous Fog-Cloud Computing. IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp 1–6
56. Ali IM, Sallam KM, Moustafa N et al (2020) An Automated Task Scheduling Model using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems. IEEE Trans Cloud Comput 1. https://doi.org/10.1109/TCC.2020.3032386
57. Hosseinioun P, Kheirabadi M, Kamel Tabbakh SR, Ghaemi R (2020) A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. J Parallel Distrib Comput 143:88–96. https://doi.org/10.1016/j.jpdc.2020.04.008
58. Jayasena KPN, Thisarasinghe BS (2019) Optimized task scheduling on fog computing environment using meta heuristic algorithms. 2019 IEEE International Conference on Smart Cloud (SmartCloud). pp 53–58
59. Ghanavati S, Abawajy J, Izadi D (2022) An Energy Aware Task Scheduling Model Using Ant-Mating Optimization in Fog Computing Environment. IEEE Trans Serv Comput 15:2007–2017. https://doi.org/10.1109/TSC.2020.3028575
60. Cloud broker. (2022, June 30). In Wikipedia. https://en.wikipedia.org/wiki/Cloud_broker. Accessed 20 Feb 2022

Publisher's Note