SCIPEDIA Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería

Revista Internacional Métodos numéricos para cálculo y diseño en ingeniería RIMNI

Balancing the load and scheduling the tasks using zebra optimizer in IoT based cloud computing for big-data applications

V. Vijayaraj[1], M. Balamurugan[1], Monisha Oberoi[2]

1 School of Computer Science of Engineering, Bharathidasan University, Tiruchirappalli, 620023, India

2 Security Services Sales, IBM Innovation Pte Ltd

Abstract

Task scheduling is one of the major problems with Internet of Things (IoT) cloud computing. The need for cloud storage has skyrocketed due to recent advancements in IoT-based technology. Sophisticated planning approaches are needed to load the IoT services onto cloud resources professionally while meeting application necessities. This is significant because, in order to optimise resource utilisation and reduce waiting times, several procedures must be properly configured on various virtual machines. Because of the diverse nature of IoT, scheduling various IoT application activities in a cloud-based computing architecture can be challenging. Fog cloud computing is projected for the integration of fog besides cloud networks to address these expectations, given the proliferation of IoT sensors and the requirement for fast and dependable information access. Given the complexity of job scheduling, it can be difficult to determine the best course of action, particularly for big data systems. The behaviour of zebras in the wild serves as the primary basis of stimulus for the development of the Zebra Optimisation Algorithm (ZOA), a novel bio-inspired metaheuristic procedure presented in this study. ZOA mimics zebras' feeding habits and their defence mechanisms against predators. Various activities are analysed and processed using an optimised scheduling model based on ZOA to minimise energy expenditures and end-to-end delay. To reduce makespan and increase resource consumption, the technique uses a multi-objective strategy. By using a regional exploratory search strategy, the optimisation algorithm may better utilise data and stays out of local optimisation ruts. The analysis revealed that the suggested ZOA outperformed other well-known algorithms. It was advantageous for big data task scheduling scenarios since it converged more quickly than other techniques. It also produced improvements of 18.43% in several outcomes, including resource utilisation, energy consumption, and make span.

OPEN ACCESS

Published: 31/05/2024

Accepted: 21/05/2024

Submitted: 04/04/2024

DOI:
10.23967/j.rimni.2024.05.009

Keywords:
Big Data Application
Internet of Things
Cloud Storage
Zebra Optimization Algorithm
Task Scheduling
Resource Utilization
big data

1. Introduction

1.1 Background of big data applications

Tools, technologies, and architectures with increased efficiency, flexibility, and resilience have emerged as a result of the Big Data age. Complex architectures with built-in scalability and optimisation capabilities are necessary for big data applications. The environments in which big data applications are deployed must be updated and upgraded on a regular maximise their scalability and flexibility [1]. Cloud-based services are being used by organisations to improve performance and reduce overall costs. Because it is lightweight, containerisation, a cloud-based technology, is becoming more and more popular. One of the most popular and widely used container-based virtualizations is Docker, which is an open-source project that makes it easy to create, operate, and deploy applications [2].

Big data applications require large-scale environments and resources in order to store, process, and analyze this massive amount of data in a distributed manner. Huge data needs can be effectively addressed by cloud computing and containerization, but accurate and appropriate load balancing is required [3]. Load balancing is crucial because the strain on servers grows exponentially as resource use rises. Furthermore, one of the key components of big data applications is the precise and quick modification of containers based on services and load.

With rising demands and use, optimising an application's performance is a constant battle. The goal of technological advancements is always to achieve greater levels of performance and efficiency. All organisations must use an environment that provides fault tolerance, performance, and dependability [4]. When it comes to providing performance, resilience, availability, and an affordable solution, cloud computing has carved out a place for itself. Cloud computing is being used by modern technologies to their advantage since it makes resources widely available in an efficient and professional way [5]. The end user now gets access to resources including software, platforms, and infrastructure without requiring any administration work thanks to the Cloud. Everything is available as a service on the cloud, including cutting-edge innovations like big data and the (IoT). As seen in Elhoseny et al. [6], several companies are providing cloud-based solutions for managing Big Data. Scaling the amount of physical resources is how elasticity is done in a multi-tiered cloud system. There are two methods for scaling resources: vertical scaling, which involves adding additional resources to the deployed virtual machines, or horizontal scaling, which involves adding more virtual machines [7]. Both approaches have more steps, have latency problems, and might be more expensive. Several paradigms and architectures have been researched and assessed in an effort to speed up procedures and optimise application development costs.

Data mining can quickly convert vast amounts of data into knowledge and ultimately value by using pertinent algorithms to process the data and uncover hidden important information [8]. Unfortunately, due to the present rapid growth of data volumes, classic techniques based on single-node serial mining are no longer appropriate for handling large amounts of data. Cloud computing, as a distributed platform, may integrate numerous computer resources and significantly boost technological capabilities. Compared to standard algorithms, it works better for processing large amounts of data [9]. Furthermore, conventional PCs may computing, which lowers the complexity and expense of cloud platform creation to some level [10]. Cloud computing models and platforms also do not have strict criteria for network nodes.

With a computing paradigm that is integrated from the advancement of traditional computers and technical networks, cloud computing has drawn more attention in recent years [11]. Cloud computing's distributed storage, virtualization, and parallel computing technologies provide fresh approaches to building computer platforms for monitoring the state of data centres' electrical equipment. Electric power companies' current basic computing facilities may be integrated to offer strong, dependable storage and processing capacity support, which is helpful for monitoring online power equipment over an extended period of time [12].

Monitoring and data gathering enhance the capacity for intelligent diagnostics and real-time analysis. Many high-reliability platforms, like Hadoop, Spark, and Storm, have developed with the fast growth of cloud computing technology, offering advantageous tools for the centralised processing of massive amounts of power equipment nursing data [13]. How can these developing computing models be integrated into the power equipment monitoring center's data processing, even if they all provide a single programming interface and hide more intricate features than conventional parallel computing programming models? It is still worthwhile to do research on the topic of combining distinct professional backgrounds to address real-world issues and various high-level applications in the monitoring system [14].

Currently, real-time traffic data analysis, weather data analysis, and medical data storage are three areas where cloud computing technology is making significant progress. Cloud computing is inexpensive, and the machines in the cluster setup don't need to meet any complex specifications. Integration with conventional data mining techniques can enable more effective administration and analysis of power monitoring data thanks to cloud computing's enormous scale and quick computation speed. In conclusion, because of its capacity to delve deeper into the shifting law of the load curve and successfully identify years [15].

1.2 Issues on energy consumption

The majority of monitoring systems were created for specific types of equipment in the early stages of condition monitoring technology development, and each scheme was dispersed and isolated. This was an info island where there was no data exchange or interaction, making it difficult to manage and thoroughly analyse monitoring data. Furthermore, it is challenging to share the hardware network, computer power, and storage—of various monitoring systems, which wastes IT resources. As a result, an integrated management system that was constructed in the main control room has surfaced and is capable of processing different monitoring data that are gathered by various monitoring devices centrally [16]. The monitoring device's present limitations, however, are that it can only transfer the streamlined, monitoring centre, and the frequency of data gathering is low. As a result, the monitoring centre will eventually gather an incredible quantity of data, and the information processing capacity of the current monitoring system will not be able to handle the demands of processing and storing such a large amount of data. It is clear that the serial processing approach has long been inadequate to handle the demands of processing massive volumes of data. Various computing challenges faced in scientific research and engineering practice have historically been attributed to the typical parallel computing paradigm based on high-performance processors.

As a result, the monitoring device processes expert data locally and feeds it to the present monitoring system. Before being uploaded, the monitoring device, for instance, has to analyse the partial discharge waveform data from high-voltage electrical equipment into the sum of discharges, matching discharge phase [17]. Uploading "familiar data" rather than "raw data" can save money on storage at monitoring centres and network transmission expenses. Even yet, a monitoring centre that combines data from several monitoring device specifications still has a difficult time diagnosing target device failure and doing a thorough condition assessment.

1.3 Issues on resource allocation

Various formulations of the resource allocation problem (RAP) have been suggested in line with various issue scenarios; the RAP may describe all real-world circumstances. Internet of Things (IoT) resource allocation problems (RAPs) are large-scale and multi-faceted, and deterministic algorithms are unable to solve them because they are nondeterministic polynomial (NP)-complete. While genetic algorithms (GA) and other NP algorithms have been researched for their ability to identify near-optimal solutions, they have a propensity to generate a significant number of infeasible keys while searching [18]. Due to GA's shortcomings, a particle swarm optimisation (PSO) metaheuristic clustering method was suggested for nonlinear MORAP. This method aims to find the Pareto-optimal keys, which are solutions that are not overshadowed by other solutions; in other words, solutions that improve one preference criterion without compromising another. To tackle scheduling challenges, a Pareto optimum solution based on time is employed to convert decisions. The suggested Pareto optimisation methods have proven to be effective in producing optimal solutions. There is an urgent need to optimise energy usage in order to extend the lifespan of the network, as the Internet of Things (IoT) is primarily used for environmental monitoring, data collecting, and processing. Due to their limited battery life, sensor and actuator nodes in the Internet of Things (IoT) may be prematurely destroyed if their resources are not used efficiently [19]. As a result, EC paradigms that revolve around edge nodes have grown in popularity as a way to address IoT's MORAP. This change in strategy has prompted a lot of research on RA as a way to handle traffic pressure and the difficulties of IoT and cloud computing. Scheduling service resources, guaranteeing quality of service (QoS), and merging multiple services are some of the well-established issues that come with the edge computing paradigm, which is related to cloud computing [20].

Research on the potential uses of big data technologies based on cloud computing in the electricity sector is now in its early stages. When it comes to online monitoring data, the majority of cloud computing systems now used by monitoring centres rely on a single Hadoop architecture, which has its limits when it comes to storing data prior to centralised processing and causes processing delays that are too lengthy [21]. Streamlining data processing to better meet the flow of data is the way of the future. The most prevalent approach involves using a database management which might not be compatible with earlier systems, and there hasn't been a complete and efficient solution that addresses velocity, volume, and diversity yet. The issue of real-time anomaly detection and the need for quick processing of monitoring data are gaining prominence [22]. Even while cloud computing's entry into the power sector is very significant from a research standpoint, the industry's use of the technology is still in its early stages, and further investigation is required before it can be integrated into power generation. Although cloud computing has garnered a lot of interest due to its great speed, no research has yet examined how to use it for processing massive amounts of data in real-time [23]. Smart city or digital twin city development with VR capabilities may be accelerated with the help of data management and multisource access technologies, which also offer robust support for intelligent on a city-scale [24]. Due to the complex process of the integrated prediction model and meeting in real-time for intelligent power schemes, the processes have often become more challenging for single computing resources to handle.

Implementing communication protocols to plan data transfer is an excellent method to reduce or eliminate data collision altogether [25]. When it comes to multitasking, the simplest and oldest scheduling protocol or algorithm is the round robin (RR) static algorithm. It provides for the equitable distribution of time slots across resources and servers. The cyclic queue, which is constrained by a time slice, also called quantum time, performs each job in turn. A real-time pre-emptive method, round robin controls a node's access at each transmission instance according to a specified circular sequence and reacts to real-time occurrences. On the other side, resource-based (RB) algorithms prioritise the allocation of resources from highest to lowest using a greedy approach and a heuristic technique. So, to maximise throughput while minimising power consumption, it repeatedly chooses the most demanding request or job and assigns it to a suitable and available server for processing [26]. For diverse requests and jobs, the RB dynamic algorithm works well by looking at resource performance records over time to determine which one is the best fit, which improves performance overall.

1.4 Contribution of the research work

The research presented here pertains to load-balancing large data applications running in containerised systems such as Docker. Based on the Docker Swarm and ZOA architecture, this paper proposes a container scheduling technique for large data applications. This article explains how to manage the workload and service discovery of large data applications using the Docker Swarm concept. In order to decrease energy costs and end-to-end latency, various activities are evaluated and processed using an efficient scheduling model. To improve the optimisation algorithm's data utilisation and prevent it from being mired in local optimisation, a regional exploratory search method is employed..

1.5 Organization of the Work

Section 1: Includes the background of big data, issues of energy consumption, introduction of resource allocation problem and contribution of the research work.

Section 2: The review of existing techniques and its issues are mentioned.

Section 3: The motivation of the proposed model with resource allocation problem is given.

Section 4: The brief explanation of the proposed model is detailed.

Section 5: The requirement of system model and its explanation is mentioned

Section 6: The result analysis and its graphical discussion are provided.

Section 7: The final contribution of the projected model is given with its future direction.

2. Related works

Foreseeing data stream frequency changes allows Sun et al. [21] to make adjustments to the grouping method based on prediction findings from a deep reinforcement learning model. In addition to efficiently managing resources, this will allow the system to swiftly adjust to fluctuations in data streams. This 1) Identify the primary causes of load skewness in distributed stream join systems and describe the application-level load balancing problem thoroughly. 2) Construct a Gated Recurrent Unit Sequence to Sequence model for forecasting the distribution of key frequencies in streams; for real-time resolution of the load imbalance issue caused by hot keys, provide a dynamic grouping approach and a feedback-based resource elasticity scaling mechanism. 3) Using the prediction model and the technique that was suggested, create an adaptive stream join system called Aj-Stream on Apache Storm. 4). Conduct comprehensive tests on a large-scale real-world dataset as well as several synthetic datasets to assess the system's performance. Experiments with static and dynamic data streams of different skewnesses show that the Aj-Stream suggested in this article maintains consistent latency and throughput. Aj-Stream showed a 22.1% improvement in system throughput and a 45.5% reduction in scheme latency while handling data streams that fluctuate regularly, in contrast to current stream-connected systems.

In order to solve this problem, Sharma et al. [22] suggested two algorithms: dynamic SDN (dSDN) and priority scheduling and congestion management (eSDN). Nevertheless, the rate of expansion of the (IoT) is uncertain and may even be exponential in the future. In order to keep up with this trend, key to manage the increasing intricacy of diverse devices and keep network latency to a minimum. As a result, we offer temporal deep Q learning for the dSDN controller in this article, which is an extension of our earlier work. An example of a self-learning reinforcement-based model is a tDQN, or Temporal Deep Q-learning Network. The tDQN agent iteratively learns to reduce network latency by improving decision-making for switch-controller mapping using a reward-punish mechanism. We have developed a method called tDQN that optimises latency and dynamic flow mapping without controllers in the best locations. In order to dynamically redirect traffic to the most appropriate controller, a multi-objective optimisation problem is developed for flow fluctuation. The tDQN achieves better throughput, loss than conventional networks, eSDNs, and dSDNs, according to comprehensive simulation findings that account for different network circumstances and traffic.

Beginning with virtualization and distributed cloud computing, Zhu [23] lays out the idea and process for implementing load balancing and suggests a better genetic load balancing algorithm. As meta-heuristic algorithms, traditional genetic algorithms might suffer from sluggish convergence. For our simulations, we relied on the free and open-source Cloudsim cloud simulation tool. When tested in a cloud computing environment, the results prove that the enhanced evolutionary algorithm outperforms the conventional one in terms of adaptability to load balancing needs and efficiency in resource utilisation.

A method for scheduling tasks based on optimisation was presented by Chandrashekhar et al. [24]. When allocating resources in the Internet of Things (IoT), the Multi-Objective Prairie Dog Optimisation (MOPDO) method takes into explanation the makespan time and the execution time as the key objectives. Virtual Machines (VMs) are efficiently resourced by the suggested MOPDOA, which selects the host with the most available resources. The search procedure will be maintained with the assistance of MOPDO to identify a suitable virtual machine (VM) for allocating resources. To schedule activities for virtual machines (VMs), the load balancing procedure must be started when resources are allocated to them. With a 100-task assignment, Particle Swarm Grey Wolf Optimisation (PSGWO) achieves a makespan time of 20s compared to MOPDOA's 12s. In a similar vein, the current Improved Multi-Objective Multi-Verse Optimizer achieves 186.33s for execution for 10 virtual machines, whereas the suggested method takes 175.45s for various VRs.

In this study, Muneeswari et al. [25] offer a new method for virtual machines in the cloud. A load balancer, which incorporates a deep network known as the Bi-LSTM approach, received input tasks from several users and forwarded them to it. When there is an imbalance in the load, the virtual machine migration will start by informing the load balancer of the specifics of the tasks. First, it equalises input loads in virtual machines (VMs). Then, it undergoes optimisation via Genetic Expression Programming (GEP). By comparing it to other methods like MVM, PLBVM, and VMIS, we were able to ascertain that the suggested LBVM is efficient according to many evaluation criteria like configuration delay, detection rate, accuracy, and so on. Compared to the current methods used by MVM, PLBVM, and VMIS, the suggested method shortens the migration time by 49%, 41.7%, and 17.8%, respectively, according to the experimental data.

The Map Reduce was created by Sundara Kumar and Mohan [26] and is an innovative and unique approach to improving the performance of data analytics. Scheduling data across cluster nodes on a larger network and putting it into distributed blocks called chunks are both handled by the Hadoop-Map Reduce paradigm. After analysing the results of several goal solutions, the best suited ones are chosen based on their short access time and high latency. The trials were run in a simulated environment using data from cluster racks and nodes located in different locations. In conclusion, the findings demonstrate a 30–35% improvement in processing speed compared to earlier techniques in big data analytics. Methods for optimising the search for optimal solutions inside a cluster of nodes, with a success rate of 24-30% when dealing with solutions of several objectives.

In order to reduce the response users and successfully protect the integrity of network communication, Saba et al. [27] devised a distributed load balancing protocol that uses particle swarm optimisation for secured data management. With the use of distributed computing, it moves expensive computations closer to the node making the request, which lowers transmission overhead and delay. In addition, the suggested approach safeguards the communication machines against harmful devices by regulated trust evaluation. When compared to other options, the suggested protocol significantly outperformed them in terms of energy success rate (17%), end-to-end latency (14%), and network cost (19% on average), according to the simulation findings.

Particle swarm optimisation (PSO), throttled load balancing, evenly spread current execution (ESCE), and round robin (RR) are the current load-balancing methods that Shahid et al. [28] has evaluated in terms of performance. This research uses a cloud analyst platform to provide a comprehensive performance review of several load-balancing methods. We also measured total cost, optimised response time (ORT), data centre processing time (DCPT), virtual machine costs, data transfer costs, and efficiency with respect to different user bases and workloads in order to determine the best configurations for service broker policies that balance virtual machines. Prior research on throttled load-balancing algorithms, round-robin execution with equally distributed current, and virtual machine efficiency and response time largely ignored the relationship and the practical relevance of the application. There has been research into comparing various load-balancing methods. Various service broker policy (SBP) experiments have been conducted to demonstrate the capabilities of the load-balancing algorithm.

Swarm Intelligence (SI) is a cloud computing load-balancing method proposed by Al Reshan et al. [29]. Load global optimisation is not taken into account by any of the various options studied in the literature, algorithm, ACO, PSO, BAT, GWO, and many more. Grey Wolf Optimisation (GWO) research. This work proposes a hybrid GWO-PSO method that combines the strengths of both quick convergence and global optimisation. The load-balancing problem may be solved by combining these two approaches, which improve system efficiency and allocation of resources. While lowering total reaction time and attaining globally optimised quick convergence, the results of this research are encouraging when compared to other conventional methodologies. Overall, the suggested method reduces reaction time by 12% compared to competing methods. In addition, the suggested GWO-PSO method enhances PSO's convergence to 97.253% using the most optimum value received from the objective function.

An algorithm that ranks jobs according to their due date for completion has been suggested by Javadpour et al. [30]. Physical devices are also grouped according to their setup condition. Moving forward, the suggested approach will distribute tasks to nearby physical machines that share the same priority class. In addition, by utilising the DVFS approach, we lessen the energy consumption of the machines that handle the low-priority jobs. If the machines' scores change, or if the workload balance is compromised, the jobs are migrated using the proposed mechanism. Using the CloudSim package, we tested and verified the suggested approach. The simulation show that the suggested strategy reduced power usage by 20% and energy consumption by 12%.

To address the limitations of timely task execution and available resources, Bebortta et al. [31] propose method to enable optimum job offloading. This method distributes resources devices. In order to alleviate the strain on limited fog resources and expedite time-sensitive activities, the offloading problem is formulated as an integer linear problem. A task prioritisation strategy is utilised to minimise latency and energy consumption of fog nodes, taking into consideration the high dimensionality of activities in a dynamic environment. The results show that compared to benchmark methods, the proposed method outperforms them in terms of service delay. In short, the suggested method improves system efficiency with regard to latency and power consumption, and it provides an effective and realistic solution to the problems caused by fog computing and the Internet of Things.

The drawbacks and research gap of the existing techniques are mentioned in the upcoming section.

2.1 Ideology from the related works

What follows is an analysis of the problems found in the recently published research on bio-inspired and current metaheuristic load balancing systems:

(i) When allocating jobs to appropriate virtual machines, most of the metaheuristic load balancing algorithms that were evaluated did not structuralize the elements of quality of service, service cost, and energy cost.

(ii) When it comes to resource allocation, the majority of the metaheuristic load balancing methods that were studied fell short when it came to creating a workable decision process that requires several load balancing criteria to cooperate.

(iii) When allocating workloads to appropriate virtual machines (VMs), the current hybrid metaheuristic loads balancing algorithms had trouble striking a balance between exploration and exploitation.

3. Resource optimisation problem and motivation

In this section, we will examine the problem formulation and the reasons behind creating the simulation in further detail.

3.1 Resource optimisation problem statement

With the (IoT), a wide diversity of tasks is possible, from monitoring sleep to keeping tabs on daily activities. Wearable and handheld devices are getting smarter and can link to social media accounts and monitor data, which improves people's lives. However, there are a lot of data transport and storage issues that come along with all this activity. A big data environment is being planned in the hopes of an interconnected world with a hundred-fold increase in user data-rate and connected devices, a tenfold increase in battery life for massive machine communiqué [32]. Problems with scalability, latency, and compatibility grow dramatically with the addition of even a single node leading to underappreciated services. Within the context of big data, there are two primary types of resource optimization problems: allocation and scheduling. There have been several attempts to optimize the Internet of Things (IoT), but most of these studies are either domain-specific or place too much emphasis on resource allocation at the expense of resource scheduling. Examples of domain-specific literature include studies devoted to healthcare job scheduling and management [33–34], transportation task offloading and scheduling [35], and industrial automations [36]. It is safe to state that there has not been complete adoption and implementation of a universal IoT framework that can be used in all IoT scenarios. With the correct strategy, optimizing resources can make docile to user expectations, which is a big step toward easing the complexity of popular IoT systems. In response to these issues, this study presents a new resource optimization technique that may be used in different Internet of Things settings.

3.2 Motivation and lapses

The Internet of Things (IoT) and value-added services generate and consume vast amounts of data; as a result, multidimensional optimization problems, such as how to choose the best service configuration in real-time and how to provide an efficient scheduling scheme for edge services, demand substantial research and development efforts [37]. Other issues with the dynamic resource allocation (DRA) method include the high IoT-based cloud systems and the resource under-utilization problem. Because of their limited storage and power capabilities, the edge nodes exacerbate the security flaws introduced by the decentralized nature of the Internet of Things' topology. An effective resource optimization strategy that gives equal weight to scheduling and resource allocation is required to avoid these mistakes. There are a number of critical areas that need a breakthrough, including the algorithms' ability to satisfy users' demanding requirements (QoE besides QoS) and the longevity of edge nodes, particularly sensor nodes.

4. Proposed methodology

As the energy consumption (EC) paradigm promotes processing data closer to the source of creation, minimizing the amount of data transfers between devices and a centralized node, it enhances security and quality assurance. Using clustering techniques appropriately cuts latency in half, allowing IoT devices to complete [38]. This paper takes ZOA, an optimization technique, a step further by incorporating clustering aspects. In order to address the question of theories are practical for managing and optimizing resources in an internet of things (IoT) setting, this paper presents an algorithm that combines elements of the EC paradigm with the zebra optimization algorithm. This algorithm can solve scheduling and resource distribution problems in the IoT, and it is relatively secure. As a result, it provides an efficient method for optimizing resources in an IoT setting.

By allocating requests/tasks a time slice/quantum time based on their size without prioritizing requests, the suggested algorithm guarantees that all requests find an economical allocation of resources, which in turn provides the best presentation in terms of regular time. Lastly, the suggested system enables the distribution of requests to available resources according to the size of resources. Optimizing resources for the Internet of Things is further enhanced by this allocation criterion. When allocating resources, the quantum time is considered. In this context, we will use the symbols Q for quantum time, TTS for the total task size that the edge nodes have allocated for execution, and TR for the transfer rate. If resource is $Q(i,j)$, the task size is $TS(i,j)$, and the transfer rate is $TR(i,j)$, then be expressed as:

$$TAT = \frac{TS(i,j)}{TR(i,j)} \times Q(i,j) \tag{1}$$

where $Q(i,j)$ is defined as:

$$Q = \frac{TS(i,j)}{TR(i,j)} \tag{2}$$

as well as TTS stands for total task size, which is the total of all tasks' (ST) awaiting resources. The authors of this work set out to fill this knowledge vacuum by creating an optimizer that mimics the hunting and defensive behaviors of zebras. The zebra is a member of the ZOA population, which is an optimizer for populations. Mathematically speaking, the plain where the zebras are problematic, and each zebra characterizes a potential solution to the tricky.

Decision variable values are based zebra is located in the search space. As a result, the problem variables may be represented by the elements of a vector, and each zebra in the ZOA can be represented by this vector. A zebra population can be expressed as a matrix. The zebras are first placed in the search space at random. Equation (3) specifies the ZOA population matrix:

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ X_N \end{bmatrix}_{N\times m} \tag{3}$$

where X is the zebra populace, X_i is the ith zebra, $x_{i,j}$ is the worth for the j-th problem variable projected by the ith zebra, N is the sum of populace members (zebras), besides m is the sum of decision variables.

The optimization issue has several potential solutions, and each zebra stands for one of them. Thus, the goals function may be assessed using the suggested the variables in the issue. In order to provide the values of the goal function as a vector, we use Eq. (4):

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ F(X_N) \end{bmatrix}_{N\times 1} \tag{4}$$

the i-th zebra's objective function value (F_i) and the vector of values (F) are defined. Finding solution to a problem is as simple as comparing the values acquired for the objective purpose; this will examine the quality of each potential solution and reveal which one is the best fit. When solving a minimization problem, the optimal solution is the answer that has the lowest objective function value. Alternatively, in maximizing issues, the optimal candidate is the one that has the highest objective function value. The ideal candidate solution needs to be found in each iteration since the zebra locations and, by extension, the values of the goal purpose, are updated throughout each iteration.

Members of the ZOA have been date by studying two zebra behaviors that occur in nature. Two examples of this kind of conduct are:

(i) Foraging, and

(ii) Defense strategies in contradiction of predators.

Therefore, in each repetition, members of the ZOA populace are updated in two dissimilar stages.

Phase 1: Foraging behavior

Initially, the population is revised according to zebra forage-hunting behavior models. Zebras typically consume a diet of grasses and sedges, although they will also eat buds, fruits, bark, roots, and leaves if their preferred foods become limited. Between sixty and eighty percent of a zebra's time is devoted to feeding, depending on the quantity and quality of grass [38]. One kind of zebra is the plains zebra. This pioneer grazer clears the way for other zebra species that rely on grasses by feeding on the canopy of taller, less nutritious grass. As the pioneer zebra, the top performer in a population guides the others to its location in the search space in ZOA. So, utilizing Eqs. (5) and (6), we can mathematically simulate the process of zebras changing their phase

$$x_{i,j}^{new,P1} = x_{i,j} + r\,.\,(PZ_j - I\,.\,x_{i,j}) \tag{5}$$

$$X_i = \begin{cases} X_i^{new,P1}, & F_i^{new,P1} \\ X_i, & else \end{cases} \tag{6}$$

where $x_{i,j}^{new,P1}$ is the new position of the ith phase, $x_{i,j}^{new,P1}$ is its jth dimension charge, $F_i^{new,P1}$ is its value, PZ is which is the dimension, r is a random sum in interval [0;1], $I = round(1 + rand)$, where $rand$ is a random sum in the intermission [0,1]. Thus, $I \in \{1, 2)$ besides if limit $I = 2$, then there are much more vicissitudes in populace movement.

Phase 2: Defense policies against predators

Phase two involves updating the search space position of ZOA population members using strategy against predator assaults. Even though lions are the most common predators of zebras, other animals such as leopards, cheetahs, brown and spotted hyenas, wild dogs, and wild dogs pose a threat as well [39]. Another animal that zaps zebras when they get near water is the crocodile. The way zebras defend themselves differs in response to different predators. In order to evade a lion's assault, a zebra would often run in a zigzag pattern or make erratic, sideways turns. When smaller predators like hyenas and dogs ambush a hunter, the zebras respond by becoming more combative. Assumption number one in the ZOA design is that the subsequent two events occur with equal likelihood:

(i) In the first scenario, the zebra anticipates an assault from a lion and plans an escape; in the second scenario, additional predators target the zebra, and it decides to take the offensive.

When lions attack zebras, the initial tactic is for the zebras to try to flee from where they are right now. Thus, this tactic may be exactly characterized by the style S_1 in Eq. (7). Second, when other predators attack a zebra, the rest of the herd will rush up to the wounded animal, forming a protective structure in an effort to confuse and terrify the assailant. Mathematically, this zebra tactic is characterized by mode S_2 in Eq. (7). In the process of repositioning zebras, a new location is considered valid if it recovers the charge of the goal function. Using Eq. (8), we can simulate this update condition:

$$x_{i,j}^{new,P2} = \begin{cases} S_1{:}\, x_{i,j} + R\,.\,(2r-1)\,.\left(1 - \frac{t}{T}\right).\,x_{i,j} & P_s \le 0.5 \\ S_2{:}\, x_{i,j} + r\,.\,(AZ_j - I\,.\,x_{i,j}), & else \end{cases} \tag{7}$$

$$X_i = \begin{cases} x_{i,j}^{new,P2}, & F_j^{new,P_2} < F_i \\ X_i, & else \end{cases} \tag{8}$$

where $x_{i,j}^{new,P2}$ is the novel status based on second phase, $x_{i,j}^{new,P2}$ is its jth dimension value, F_i^{new,P_2} is its objective function charge. After the first two phases of a ZOA iteration, the population members are updated to finish the iteration. Updates to the algorithm population are carried out in accordance with Eqs. (5) to (8) until the algorithm is fully implemented. Over the course of several rounds, the optimal candidate solution is refined and stored.

Phase 3: Computational complexity

Here we take a look at how complicated ZOA is to compute. When preparing to use ZOA, the time complexity is $O(N \cdot m)$, where N is the total sum of zebras and m is the sum of variables in the problem. ZOA incorporates the iteration count T such that every member of the population undergoes bi-phase updates and objective function evaluations in each iteration. Computing this update procedure has complexity of $O(2 \cdot N \cdot m, \cdot T),$. Therefore, ZOA's overall computational complexity is $O(N \cdot m \cdot (1 + 2 \cdot T))$. The roadmap of the projected perfect is exposed in Figure 1.

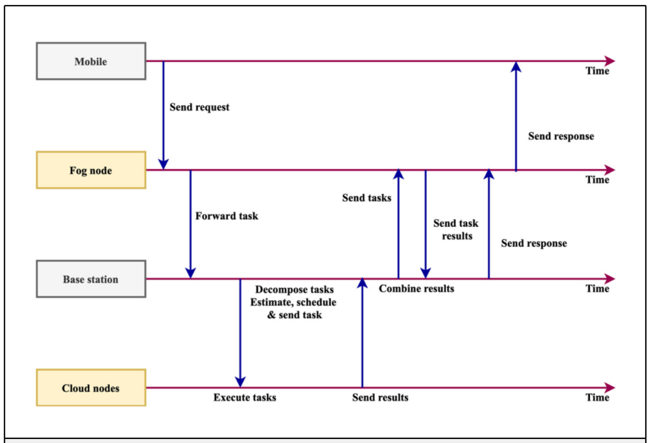

Figure 1. Workflow of the projected scheduling scheme

The planned pipeline has been dissected and its process is shown in Figure 1. Big data environment mobile user devices, base stations, cloud nodes, and fog nodes make up this system. The scheduler's steps for the suggested strategy are as follows: (1) A appeal is sent to the fog node with the mobile data. This (2) creates the fog node and sends the job to the main station. (3) The base station estimates, schedules, and assigns tasks to the nodes in the cloud. (4) At a certain interval, the cloud node will submit the completed job. Users start the process, and nodes in the cloud and fog keep in touch with each other. The job that was passed from the fog node is collected by the station. After receiving the job, the base station breaks it down into its component parts, makes an estimate, and then provides the aggregate result. The findings are combined by the base station once the work is decomposed. The task is then scheduled and sent. At a certain moment, the aggregated outcome is applied in the base station. Users get the final, combined result when it is computed. The request is sent to the fog node using the suggested architecture, which uses the smartphone network. After leaving the fog node, the job is directed to station. Through the base station, the job is sent and received. The base station sends the deconstructed job to the cloud node. The work is done in the cloud node, and then the output is performed. The reconfiguration agent comes after the architectural step.

4.1 Reconfiguration agent

When the judgment unit uses the most recent state characteristics to create a reconfiguring activity. At each T step, the RA reaps the advantages of both the prior reconfiguring activity and the transactions that came before it. Rearranging the reconfiguring action is done using the reconfiguration agent. Using the reconfiguration agent, the reconfiguring procedure may be rearranged. Here we will go over the steps to create the reward, training, and state components.

4.1.1 Reward

Keep in mind that reducing the total delay cost for all network jobs is the objective of the problem being researched. Reducing the total cost of delay should be the goal of any reconfiguration effort. What this means is that every reconfiguration procedure has to keep the total cost of configuration step to a minimum. Maximizing the cumulative reward for an episode is the goal of a deep learning agent. So, the compensation for each reconfiguration phase is defined as the inverse of the tardy was just applied per second at that stage.

In one stage, employing both buffer jobs and finished tasks, the timeliness cost is generated. During a reconfiguration process, allow each completed job to be transformed. When a present finished. R' besides r'' are the starting and end duration of the contemporary reconfiguration stage, singly. Consequently, the period *step* $[r', r'']$ reward is exposed in Eq. (9):

$$R_R = \frac{1}{r'} - \frac{1}{r''}\left[\sum_{x=0}^{N}\sum_{y=0}^{O}\frac{\propto_y}{C_y}(r' - \max(r', l_y))\right] + \left[\sum_{y=0}^{N}\frac{\propto_y}{C_y}(r' - \max(r', l_y))\right] \tag{9}$$

The first and ultimate stages of reconfiguration are denoted as r' and r'', respectively. The reconfiguration judgment awards and distributes the cumulative prizes from each episode. What follows is an explanation of the reconfiguration judgment.

4.1.2 Reconfiguration judgment

To decide whether to reorganize, the reconfiguration judgment is activated when the initial device finishes a job. Only when the RA wants to decide to reconfigure does it do so. With the help of the reconfiguration judgment system, human reconfiguration is reduced in frequency. In any case, the RA will need a lot of practice events before it figures out how to make reconfigurations less frequent.

Let $t' = 0, 1, 2, \cdot\cdot\cdot, N$ characterize the existing manufacture mode and $w_{t'}$ signify its buffer. The present scheme time is designated by t', r^c. The x's current area tardy cost is represented by β_x, where the moment in time is defined by current cost. The Eq. (10) represents the current cost

$$\beta_x = \begin{cases} \propto_y & r^c > l_y \\ \frac{\propto_y}{C_y} & r' > l_y > r'' \\ 0 & else \end{cases} \tag{10}$$

At the three main decision unit makes a decision to represented as r^c, queue is represented as l_y. The cost function is signified as r' and r''. A combination of the task's beginning and end values, as well as its current computation cost, yields the final result.

Case 1: When $w_{t'}$ is blank.

Case 2: When β_y is slighter than the xth % unresolved task in $w_{t'}$ here, β is a list β_y is the mean of the contemporary unit y

$$\beta_y = \frac{1}{N}\sum_{x=0}^{N}\beta_x \tag{11}$$

The mean spoken as β_x, and the entire sum of examples obtainable is represented as N.

Case 3: When $w_{t'}$ and N additional tasks have been processed and β_y is less than y% of β. Pay attention to the fact that every attribute of a state is an array. To accurately reproduce the qualities of the underlying data, the standard error, mean, minimum, and maximum are calculated for each state attribute. Hence, the state vector is 16 dimensions long (4×4). We use min-max normalizing for state attributes. Equation (12) expresses the inverse of state feature $F_{t''}, F_{t''}$.

$$F_{t''} = \frac{F_{t'} - \max(F_{t'})}{\max(F_{t'}) - \min(F_{t'})} \qquad (12)$$

The functions are characterized as $\min(\cdot)$. The collection of altogether $F_{t'}$ without resampling aeras is characterized by $F_{t''}$ in this circumstance. For the reserve $F_{t''}$, the max-min algorithm gives advanced importance to task $F_{t''}$ than tasks. With max-min, numerous minor processes may operate in tandem with the bigger ones. Finishing the longest work first will establish the total timeframe [40]. A reduction in cost delay episode earns RA as a reward. Because of the reconfiguration decision, human configuration is diminished. Deep learning employs an effective preparation agent to plan the job after the device reconfiguration phase is over. The scheduling agent's job is to pick the best resource and service combination for each new job that comes up.

5. System requirements analysis

The computerized large data processing system's primary features are, among other things, the ability to access data sources, process data streams, and support for individualized data processing rules. Separate introductions to these modules are forthcoming.

Access the data source

With this system, you may obtain data from a wide range of sources, the majority of which are online but also include offline options. Some applications need processing both online and modest amounts of offline data, even if the system is designed to process data streams online. The data will be stored offline in HDFS by the system. The system organizes and categorizes data stored online according to topics; users have the option to enable data loss in order to enhance application performance. Lastly, data goes into the following processing data Low once a job to a data source.

Data stream processing

Different use cases call for different data stream processing logic. The system may now accommodate many types of data processing logic thanks to a functional component that summarizes typical data processing procedures. As a result, all the user has to do is combine the necessary functional components in a flexible way and indicate their topological connection.

Data storage: Procedures for data streams to persist. Back both classic MySQL and key-value HBase databases.

Data statistics: Produce broad statistics regarding the data flow. Sum, count, average, and max/min are the aggregation functions that are supported.

Data collection: Data stream matching, data collection from the field, and output should be done periodically. In addition, you may select the sorts of fields and give them names.

Data Filtering: Restrict the flow of data. By implementing logical operations (such OR besides AND) on the consequences of corresponding several fields, filtering rules can reveal data that fits the constraints. These operations include regular, range, precise, and fuzzy matching for a field.

According to Table 1, the experimental setup consists of four Swarm nodes: one master node and three worker nodes. Using the NGINX service, the Swarm instructions are executed on the master node. Schedules, DNS service discovery, scalability, and container load balancing are all handled by Swarm itself on all nodes. Each node in the Swarm will have its own copy of the load balancer, which will distribute requests among them as needed.

Table 1. Swarm services for big data request

Type	Port	Container
Load Balancer	80	NGINX
Front End PHP service	8080	Apache
API server (Python)	5000	Python
API server (Redis)	6379	Redis

6. Result and discussion

The details of the projected cloud environment are detailed in the simulated environment, and the performance matrices cover each key matrix. Present scheduling and load-balancing methods are contrasted with the suggested method in the comparative analysis section. The jobs utilized in this project range from one hundred to five hundred. Figures 2 to 6 demonstrate comparisons between the suggested model and many current methodologies, including the Butterfly optimization Algorithm (BOA), the Stray Lion optimization Algorithm (SLOA), and the mother optimization Algorithm (MOA).

Figures 2 through 6 illustrate the performance of various parametric metrics in relation to distinct tasks. The analysis of the BOA model yielded the following results for 100 tasks: makespan = 482, energy consumption = 51.6157, balanced CPU utilization = 0.0168, optimized memory utilization = 5313.61, and prioritization = 5930. Subsequently, the 200th task was executed, with a duration of 1091 iterations, an energy consumption of 66.6182 ± 0.02087, optimized memory usage of 5404.47, and prioritization of 29,800. Subsequently, the 300th task was executed, with a duration of 1788 seconds, an energy consumption of 81.9739, balanced CPU utilization of 0.0202, optimized memory utilization of 5473.9, and prioritization of 44,850. The 400th task was completed with a duration of 2902 instructions, an energy consumption of 84.1332, balanced CPU utilization of 0.0935, optimized memory utilization of 5619.42, and prioritization of 49,800. Following that, the 500th task was executed with a duration of 2457 iterations, an energy consumption of 96.9864, balanced CPU utilization of 0.0126%, optimized memory utilization of 5721.11, and prioritization of 52,750.

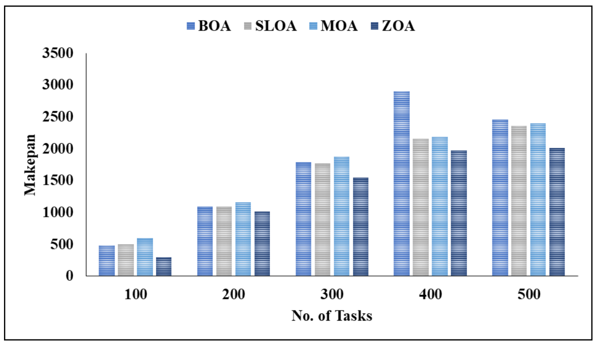

Figure 2. Visual representation of different models

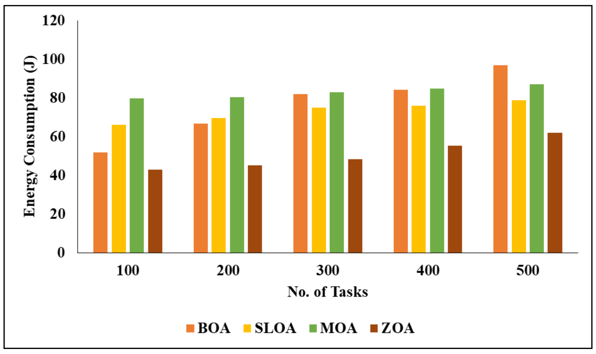

Figure 3. Graphical description of various models

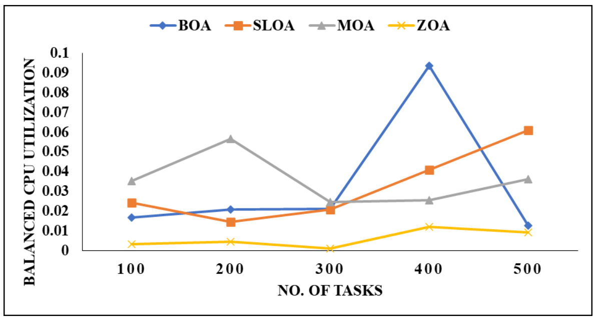

Figure 4. Analysis of different optimization models

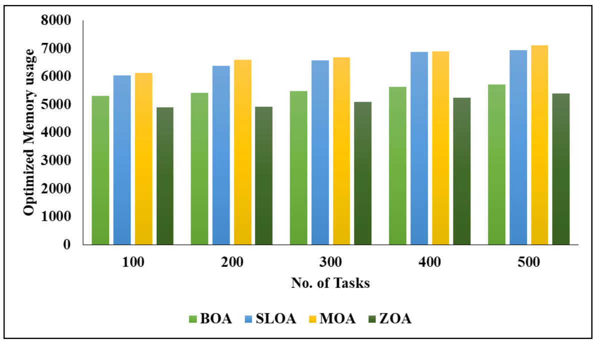

Figure 5. Visual presentation of proposed model with existing procedures

Subsequently, the SLOA model assigns the following values to the 100 tasks: span (100,506, 66.1581), balanced CPU utilization (0.02424), optimized memory utilization (6029.24), and prioritization (5950). The 200th task had a duration of 1097 iterations, an energy consumption of 69.6314, a balanced CPU utilization of 0.01473, optimized memory utilization of 6380.03, and prioritization of 30,900. The 300th task had a duration of 1769 iterations, 74.8653 0.02096 energy consumption, and balanced CPU utilization of 6573.92 45,850 correspondingly. Following that, the 400th task was executed with a duration of 2154, an energy consumption of 75.9833, balanced CPU

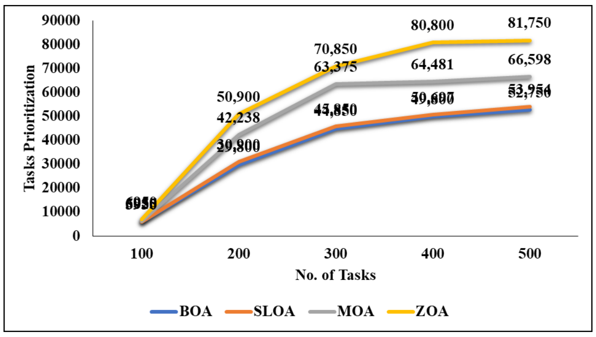

Figure 6. Analysis of different optimization models for task prioritization

utilization of 0.04096, optimized memory utilization of 6879.92, and prioritization of 50,697. Following that, the 500th task was executed with a duration of 2365, a balanced CPU utilization of 78.7313, an energy consumption of 0.06096, an optimized memory usage of 6943.92, and a prioritization of 53,954.

The MOA model then computes the makespan for the 100th task to be 100 and the balanced CPU utilization to be 601 79.8525 0.03542 6125.37 6025. Then, the 200th task was executed with a duration of 1165, an energy consumption of 80.3654 0.05649, and a balanced CPU utilization of 6596.32 42,238. Subsequently, the 300th task was executed with a duration of 1874 82.9542, optimized memory usage of 0.02465, balanced CPU utilization of 6685.68, and prioritization of 63,375. Then, the 400th task was executed with a duration of 2187, an energy consumption of 84.743 x 0.02546 x 6896.74, and a balanced CPU utilization of 64,481. Then the 500th task, with the following priorities: optimized memory usage (7098.32), makespan (2396 x 86.9845 x 0.03621), and prioritization (66,598).

The ZOA model then calculates the makespan for the 100th task to be 100 299 42.7248 0.00338 4893.05 and the balanced CPU utilization to be 6950. Then, the 200th task, with a duration of 1013 45.1718, balanced CPU utilization of 0.00461, optimized memory utilization of 4915.5, and prioritization of 50,900, was executed. The 300th task was completed with a duration of 1546 iterations, an energy consumption of 48.2737, balanced CPU utilization of 0.00125, optimized memory utilization of 5085.77, and prioritization of 70,850. Following that, the 400th task was executed, with a duration of 1972, an energy consumption of 55,4123, balanced CPU utilization of 0.01240, optimized memory utilization of 5241.27, and prioritization of 80,800. Following that, the 500th task was executed, with a duration of 2015, an energy consumption of 61.9782 0.00934, an optimized memory usage of 5383.95, and a prioritization of 81,750. The proposed model explains the results in terms of makespan, energy consumption, task priortitization, memory usage and balanced CPU utilization. In future work, the results will be improved by adding resource utllization, number of overloaded tasks, training time reduction, etc.

7. Conclusions and findings

Using ZOA to provide well-informed options for job situations and reconfiguration, this article explored intelligent planning task delivery. The problem is solved by providing a system design that allows for careful planning and reconfiguration in a big data setting. We create a quantitative approach to lower the total cost of timeliness for all jobs. Furthermore, a model for optimization is proposed that incorporates a timetable and agents that may be reconfigured. The agency's characteristics, operations, and incentives are aimed at the two services. Using cloud-edge computing in a big data setting, it claims to provide a high-performance computing strategy that makes good use of available resources and evenly distributes processing loads. The suggested protocol also utilized the ZOA method to discover effective keys with the best possible local besides global fitness functions. In order to maintain a steady connection with little latency, the fitness function makes use of the energy and reliability aspects of the channel. On top of that, network edges protect data storage and gathering from network risks while small hardware devices are dispersed and limited incur little costs. Based on the results of the experiments, it is evident that the ZOA reduces energy usage by up to 18.74J compared to other models like MOA, SLOA, and BOA. However, the proposed model achieved low performance in minimizing the makespan due to high interia weight of the proposed zebra optimization algorithm. This must be addressed in future work and also tries to minimize the high computational time.

7.1 Future Work

The future of this project depends on investigating and resolving issues related to its computational complexity and cost efficiency. Implementing the provided methodologies for Big Data requests multi-cloud surroundings is also within the realm of possibility.

Conflicts of interest: None

Submission declaration and verification: The work described has not been published previously.

Authors' contribution statement

Vijayaraj Veeramani conceived of the presented idea. Mr. Vijayaraj Veeramani developed the theory and performed the computations. Dr. M. Balamurugan – Professor, School of Computer Science of Engineering, Bharathidasan University, Tiruchirappalli and Dr. Monisha Oberoi – Director Security Services Sales, IBM Innovation Services Pte Ltd. Verified the analytical methods. All authors discussed the results and contributed to the final manuscript.

References

[1] Abualigah L., Diabat A., Elaziz M.A. Intelligent workflow scheduling for big data applications in IoT cloud computing environments. Cluster Computing, 24(4):2957-2976, 2021.

[2] Basu S., Karuppiah M., Selvakumar K., Li K.C., Islam S.H., Hassan M.M., Bhuiyan M.Z.A. An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment. Future Generation Computer Systems, 88:254-261, 2018.

[3] Bouhouch L., Zbakh M., Tadonki C. Online task scheduling of big data applications in the cloud environment. Information, 14(5):292, 2023.

[4] Rjoub G., Bentahar J., Wahab O.A. BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments. Future Generation Computer Systems, 110:1079-1097, 2020.

[5] Nguyen B.M., Thi Thanh Binh H., The Anh T., Bao Son D. Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment. Applied Sciences, 9(9):1730, 2019.

[6] Elhoseny M., Abdelaziz A., Salama A.S., Riad A.M., Muhammad K., Sangaiah A.K. A hybrid model of internet of things and cloud computing to manage big data in health services applications. Future Generation Computer Systems, 86:1383-1394, 2018.

[7] Li X., Wang L., Abawajy J.H., Qin X., Pau G., You I. Data-intensive task scheduling for heterogeneous big data analytics in IoT system. Energies, 13(17):4508, 2020.

[8] Jalalian Z., Sharifi M. A hierarchical multi-objective task scheduling approach for fast big data processing. The Journal of Supercomputing, 78(2):2307-2336, 2022.

[9] Islam T., Hashem M.M.A. Task scheduling for big data management in fog infrastructure. In 2018 21st International Conference of Computer and Information Technology (ICCIT), IEEE, pp. 1-6, 2018.

[10] Cai X., Geng S., Wu D., Cai J., Chen J. A multicloud-model-based many-objective intelligent algorithm for efficient task scheduling in internet of things. IEEE Internet of Things Journal, 8(12):9645-9653, 2020.

[11] Abohamama A.S., El-Ghamry A., Hamouda E. Real-time task scheduling algorithm for IoT-based applications in the cloud–fog environment. Journal of Network and Systems Management, 30(4):1-35, 2022.

[12] Boveiri H.R., Khayami R., Elhoseny M., Gunasekaran M. An efficient Swarm-Intelligence approach for task scheduling in cloud-based internet of things applications. Journal of Ambient Intelligence and Humanized Computing, 10:3469-3479, 2019.

[13] Rjoub G., Bentahar J., Wahab O.A., Bataineh A. Deep smart scheduling: A deep learning approach for automated big data scheduling over the cloud. In 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, pp. 189-196, 2019.

[14] Priyanka E.B., Thangavel S., Meenakshipriya B., Prabu D.V., Sivakumar N.S. Big data technologies with computational model computing using Hadoop with scheduling challenges. In Deep Learning and Big Data for Intelligent Transportation: Enabling Technologies and Future Trends, pp. 3-19, 2021.

[15] Xu X., Liu Q., Luo Y., Peng K., Zhang X., Meng S., Qi L. A computation offloading method over big data for IoT-enabled cloud-edge computing. Future Generation Computer Systems, 95:522-533, 2019.

[16] Abd Elaziz M., Abualigah L., Attiya I. Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. Future Generation Computer Systems, 124:142-154, 2021.

[17] Narman H.S., Hossain M.S., Atiquzzaman M., Shen H. Scheduling internet of things applications in cloud computing. Annals of Telecommunications, 72:79-93, 2017.

[18] Al-Turjman F., Hasan M.Z., Al-Rizzo H. Task scheduling in cloud-based survivability applications using swarm optimization in IoT. In The Cloud in IoT-enabled Spaces, CRC Press, pp. 1-32, 2019.

[19] Li C., Zhang Y., Luo Y. Neighborhood search-based job scheduling for IoT big data real-time processing in distributed edge-cloud computing environment. The Journal of Supercomputing, 77:1853-1878, 2021.

[20] Lu Z., Wang N., Wu J., Qiu M. IoTDeM: An IoT big data-oriented MapReduce performance prediction extended model in multiple edge clouds. Journal of Parallel and Distributed Computing, 118:316-327, 2018.

[21] Sun D., Zhang C., Gao S., Buyya R. An adaptive load balancing strategy for stateful join operator in skewed data stream environments. Future Generation Computer Systems, 152:138-151, 2024.

[22] Sharma A., Balasubramanian V., Kamruzzaman J. A temporal deep Q learning for optimal load balancing in software-defined networks. Sensors, 24(4), 1216, 2024.

[23] Zhu F. Cloud computing load balancing based on improved genetic algorithm. International Journal of Global Energy Issues, 46(3/4):191-207, 2024.

[24] Chandrashekhar A.S., Chandrashekarappa N.M., Hanumanthagowda P.B., Bongale A.M. Multi objective prairie dog optimization algorithm for task scheduling and load balancing. International Journal of Intelligent Engineering & Systems, 17(2):585-594, 2024.

[25] Muneeswari G., Madavarapu J.B., Ramani R., Rajeshkumar C., Singh C.J.C. GEP optimization for load balancing of virtual machines (LBVM) in cloud computing. Measurement: Sensors, 33, 101076, 2024.

[26] Sundara Kumar M.R., Mohan H.S. Improving big data analytics data processing speed through map reduce scheduling and replica placement with HDFS using genetic optimization techniques. Journal of Intelligent & Fuzzy Systems, 46(1):1-20, 2024.

[27] Saba T., Rehman A., Haseeb K., Alam T., Jeon G. Cloud-edge load balancing distributed protocol for IoE services using swarm intelligence. Cluster Computing, 26(5):2921-2931, 2023.

[28] Shahid M.A., Alam M.M., Su'ud M.M. Performance evaluation of load-balancing algorithms with different service broker policies for cloud computing. Applied Sciences, 13(3), 1586, 2023.

[29] Al Reshan M.S., Syed D., Islam N., Shaikh A., Hamdi M., Elmagzoub M.A., Talpur K.H. A fast converging and globally optimized approach for load balancing in cloud computing. IEEE Access, 11:11390-11404, 2023.

[30] Javadpour A., Sangaiah A.K., Pinto P., Ja'fari F., Zhang W., Abadi A.M.H., Ahmadi H. An energy-optimized embedded load balancing using DVFS computing in cloud data centers. Computer Communications, 197:255-266, 2023.

[31] Bebortta S., Tripathy S.S., Modibbo U.M., Ali I. An optimal fog-cloud offloading framework for big data optimization in heterogeneous IoT networks. Decision Analytics Journal, 8, 100295, 2023.

[32] Thirumalraj A., Asha V., Kavin B.P. An Improved Hunter-Prey Optimizer-Based DenseNet Model for Classification of Hyper-Spectral Images. In AI and IoT-Based Technologies for Precision Medicine, IGI global, pp. 76-96, 2023.

[33] Shafique K., Khawaja B.A., Sabir F., Qazi S., Mustaqim M. Internet of Things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios. IEEE Access, 8:23022-23040, 2020.

[34] Hameed Abdulkareem K., Awad Mutlag A., Musa Dinar A., Frnda J., Abed Mohammed M., Hasan Zayr F., Lakhan A., Kadry S., Ali Khattak H., Nedoma J. Smart healthcare system for severity prediction and critical tasks management of COVID-19 patients in IoT-Fog computing environments. Comput. Intell. Neurosci., 2022:5012962, 2022.

[35] Lakhan A., Mohammed M.A., Elhoseny M., Alshehri M.D., Abdulkareem K.H. Blockchain multi-objective optimisation approach-enabled secure and cost-efficient scheduling for the Internet of Medical Things (IoMT) in fog-cloud system. Soft Computing, 26:6429-6442, 2022.

[36] Alatoun K., Matrouk K., Mohammed M.A., Nedoma J., Martinek R., Zmij P. A novel low-latency and energy-efficient task scheduling framework for Internet of medical things in an edge fog cloud system. Sensors, 22, 5327, 2022.

[37] Lakhan A., Mohammed M.A., Garcia-Zapirain B., Nedoma J., Martinek R., Tiwari P., Kumar N. Fully homomorphic enabled secure task offloading and scheduling system for

transport applications. IEEE Trans. Veh. Technol., 71:12140-12153, 2022.

[38] Trojovská E., Dehghani M., Trojovský P. Zebra optimization algorithm: A new bio-inspired optimization algorithm for solving optimization algorithm. IEEE Access, 10:49445-49473, 2022.

[39] Caro T., Izzo A., Reiner R.C., Walker H., Stankowich T. The function of zebra stripes. Nature Commun., 5(1):1-10, 2014.

[40] Pal S., Jhanjhi N.Z., Abdulbaqi A.S., Akila D., Alsubaei F.S., Almazroi A.A. An intelligent task scheduling model for hybrid internet of things and cloud environment for big data applications. Sustainability, 15(6), 5104, 2023.