I. J. Engineering and Manufacturing, **2024, 3, 1-15**
Published Online on June 8, 2024 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijem.2024.03.01

HKCHB: Meta-heuristic Algorithm for Task Scheduling and Load Balancing in Cloud-fog Computing

Mahmoud Moshref*
Palestine Technical University -Kadoorie/ Department of Computer Science/ Faculty of Information Technology, Tulkarm, P.O Box 305, Palestine
E-mail: moshref2008@gmail.com, mahmoud.moshref@ptuk.edu.ps
ORCID iD: https://orcid.org/0000-0001-6325-8170
*Corresponding author

Sherin Hijazi
Palestine Technical University -Kadoorie/ Department of Computer Science/ Faculty of Information Technology, Tulkarm, P.O Box 305, Palestine
E-mail: s.hijazi@ptuk.edu.ps, sherinhijazi@yahoo.com
ORCID iD: https://orcid.org/0000-0003-2411-5681

Azzam Sleit
The University of Jordan/ Department of Computer Science/ King Abdullah II School for Information Technology, Amman, 11941, Jordan
E-mail: azzam.sleit@ju.edu.jo
ORCID iD: https://orcid.org/0000-0003-3383-6253

Ahmad Sharieh
The University of Jordan/ Department of Computer Science/ King Abdullah II School for Information Technology, Amman, 11941, Jordan
E-mail: sharieh@ju.edu.jo
ORCID iD: https://orcid.org/0000-0002-0290-2468

Received: 16 January, 2024; Revised: 15 February, 2024; Accepted: 17 March, 2024; Published: 08 June, 2024

Abstract: Cloud-fog computing has emerged as the contemporary approach for processing and analyzing Internet of Things applications due to its ability to offer remote resources. Cloud fog computing technology provides shared resources, information, and software packages, supporting distributed parallel systems in an open environment. It constructs and manages virtual machines to enhance efficiency and attractiveness. We have consistently strived to tackle challenges affecting the efficiency of cloud fog computing, including ineffective resource utilization and response times. The improvement of these challenges can be achieved through effective task scheduling and load balancing between Virtual Machines, this problem considered as NP-hard problem. This paper proposes a Hybrid K-means Clustering Honey Bee algorithm (HKCHB) to cluster Virtual Machines into two or more clusters. Subsequently, the hybrid Honey Bee algorithm is employed for task scheduling, enhancing load balance performance. The proposed algorithm is compared with other task scheduling and load balancing algorithms, including Round Robin, Ant Colony, Honey Bee, and Particle Swarm Optimization Algorithm, utilizing the CloudSim Simulator. The results demonstrate the superiority of the proposed algorithm, yielding the lowest response time. Specifically, the response time is reduced by 22.1%, and processing time is reduced by 47.9%, while throughput is increased by 95.4%. These improvements are observed under the assumption of multiple tasks in a heterogeneous environment, utilizing one or two Data Centers with Virtual Machines**.** This contribution suggests that network systems utilizing the Internet of Things and cloud fog computing will advance in the future to operate highly efficiently within real-time system frameworks.

Index Terms: Cloud computing; Fog computing, Internet of Things, Task scheduling; Load Balance; Honey Bee; Clustering; k-Means.

1. Introduction

Cloud computing delivers distributed services on demand, provided by major service providers like Amazon, Google, and Microsoft [1]. These services, such as Infrastructure-as-a-Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS), are made accessible to end-users through the Internet. However, cloud computing faces challenges like centralization, low performance, latency, single-point failure, and security concerns. In response, the concept of fog computing aims to reshape the landscape created by cloud computing, decentralizing and localizing data-centric clouds [2]. Fog computing operates by bringing the processing, analysis, and storage of data generated by Internet of Things (IoT) devices closer to the edge. Instead of transferring information to the cloud server, these tasks are performed locally at the fog node [3, 4]. The collaboration of cloud and fog computing has become a significant evolution in distributed systems, addressing issues in current Internet of Things (IoT) architectures [4, 5]. This combined approach, known as cloud-fog computing, is defined as a model that offers convenient and on-demand access to a shared pool of resources, including networks, storage, services, and applications, requiring minimal effort from users [6, 7]. This heterogeneous environment provides a rapid and on-demand array of services to end-users.

Cloud-fog computing enables users or devices, including machines, to access software applications without the need for installation. End-users or devices can utilize services, software, or hardware infrastructure on demand. The distinct advantages of Cloud-fog over other technologies lie in its ease of management, cost reduction, uninterrupted services, enhanced agility, effective disaster management, green computing practices, and minimal management effort [8].

Task scheduling with load balancing stands out as a critical factor influencing cloud performance, primarily due to its ability to prevent clients from directly accessing back-end servers, offering security advantages by concealing the inner network's structure. This issue falls within the category of NP-hard problems due to the extensive solution space, making it time-consuming to find an optimal solution. Unfortunately, there are no algorithms capable of producing optimal solutions within polynomial time for this problem. Therefore, an efficient task scheduling algorithm becomes crucial to adhere to virtualization principles and demand. The primary objective of a task scheduling algorithm is to enhance system throughput, achieve load balance, and reduce completion periods. To meet these objectives, meta-heuristic-based algorithms are employed, aiming to achieve near-optimal solutions [9, 10, 11].

The primary objective of this research paper is to evaluate the performance of the HKCHB algorithm in comparison to task scheduling and load balancing algorithms within a simulated cloud fog computing environment. The evaluation will be based on three metrics: response (execution) time, processing time, and throughput. The findings of this research will contribute to the enhancement of scheduling and load balancing strategies in cloud fog computing and highlight the potential of nature-inspired optimization algorithms to improve the efficiency and scalability of distributed systems.

To advance the state of the art in task scheduling services within cloud fog computing environments, the following improvements are proposed:

- The adoption of a hybrid meta-heuristic method named HKCHB, which is inspired by nature, to achieve efficient task scheduling and load balancing in cloud fog computing environments.
- The utilization of three metrics—response (execution) time, processing time, and throughput—to demonstrate the effectiveness of the proposed model for cloud fog computing systems.
- A comparative analysis of the proposed scheduling strategy's response (execution) time, processing time, and throughput against those of other approaches when applied to realistic workloads, aiming to facilitate the future practical application of this algorithm.

The proposed algorithm, denoted as HKCHB, is implemented using the Java programming language and tested on the CloudSim simulator. Performance comparisons are conducted with other algorithms, including Round Robin (RR), Ant Colony (ACO), Particle Swarm Optimization Algorithm (PSO), and Honey Bee (HB). All these algorithms are also implemented in Java, and the proposed HKCHB algorithm exhibits superior performance, particularly in terms of response time.

The structure of this research is organized as follows: Section 2 provides a literature review of related work addressing task scheduling and load balancing problems. Section 3 outlines the research methodology, illustrating the cloud-fog model and presenting task scheduling algorithms. Section 4 details the experiments and results evaluation. Finally, Section 5 concludes the research.

2. Literature Review

Scheduling tasks in a heterogeneous environment poses a significant challenge, particularly in cloud-fog environments where traditional scheduling algorithms often fall short due to their limited adaptability to changing conditions. Meta-heuristic algorithms have emerged as a solution to enhance task scheduling performance in such

dynamic environments. This literature review provides an overview of studies utilizing meta-heuristic algorithms to address task scheduling challenges.

Researchers Mohamed Abd Elaziz, Laith Abualigah, and Ibrahim Attiya [4] proposed an alternative task scheduling technique for IoT requests in a cloud-fog environment. They introduced a modified artificial ecosystem-based optimization (AEOSSA), incorporating operators from the Salp Swarm Algorithm (SSA). AEOSSA demonstrated superior performance in comparison to other well-known meta-heuristic methods.

M. Santhosh Kumar, K. Ganesh Reddy, and Rakesh Kumar Donthi [5] suggested incorporating the Shark Search Krill Herd Optimization (SSKHOA) method into cloud-fog computing's task scheduling. The SSKHOA algorithm, combining the shark search and krill herd algorithms, exhibited efficient global and local search capabilities, outperforming traditional task scheduling techniques.

Tran Cong Hung and Nguyen Xuan Phi [6] conducted a study to assess the impact of specific parameters on the performance of load balancing in Cloud Computing. These parameters, including Million Instructions Per Second (MIPS) and the Length of the Cloudlet, were investigated using the CloudSim simulator. The key finding of their research indicates that the runtime parameter is the most significantly affected when modifying MIPS and the length of the cloudlet.

Ali Al-maamari and Fatma A. Omara [9] implemented a Dynamic Adaptive Particle Swarm Optimization algorithm (DAPSO) to optimize task runtime, minimizing makespan, and maximizing resource utilization. DAPSO demonstrated enhanced performance compared to the basic PSO algorithm.

Kousik Dasgupta, Brototi Mandal, and others [12] proposed a load balancing strategy using Genetic Algorithm (GA) to balance the load in cloud infrastructure while minimizing makespan. Their algorithm outperformed existing approaches like First Come First Serve (FCFS) and Round Robin (RR) in simulations.

Safwat A. Hamad and Fatma A. Omara [13] introduced a task scheduling algorithm utilizing Genetic Algorithm (GA) to allocate and execute application tasks. The primary objective of this algorithm is to minimize both task completion time and cost while maximizing resource utilization. The performance of their proposed algorithm was assessed using the CloudSim toolkit.

Ren Gao and Juebo Wu [14] presented a load balancing approach using ant colony optimization (ACO) to dynamically balance workload in a cloud computing platform. The proposed strategy, incorporating forward-backward ant mechanisms and max-min rules, demonstrated efficient dynamic load balancing with high network performance.

A.I.Awad, N.A.El-Hefnawy, and others [15] introduced a mathematical model, Load Balancing Mutation Particle Swarm Optimization (LBMPSO), considering reliability and availability in cloud computing environments. LBMPSO demonstrated savings in makespan, execution time, round-trip time, and transmission cost.

Baris Yuce, Michael S. Packianather, and others [16] described the Bees Algorithm, inspired by honey bees' natural foraging behavior, for optimizing benchmark functions. They implemented and analyzed different versions of the algorithm for various optimization tasks.

Duc Truong Pham and Marco Castellani [17] conducted an experimental investigation into the performance of the Bees Algorithm, analyzing its strengths and weaknesses. The study unveiled the most suitable parameterizations for different optimization tasks. The algorithm was tested on four real-world minimization tasks, demonstrating the effectiveness and robustness of the bee's foraging metaphor, especially on complex and high-dimensional fitness landscapes.

In a related context, Bashar Al-Shboul and Sung-Hyon Myaeng [18] proposed two algorithms, Genetic Algorithm Initializes KM (GAIK) and KM Initializes Genetic Algorithm (KIGA), to address the initialization problem. To showcase the effectiveness and efficiency of their algorithms, they conducted a comparative study involving GAIK, KIGA, Genetic-based Clustering Algorithm (GCA), and Fuzzy C-Means (FCM). The results indicated that KIGA outperformed the others, achieving high clustering accuracy.

F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah [19] proposed the Multi-Objectives Grey Wolf Optimizer (MGWO) algorithm to reduce delay and energy consumption in the fog broker. Simulation results validated the effectiveness of MGWO compared to state-of-the-art algorithms.

H. K. Apat, B. Sahoo, V. Goswami, and R. K. Barik [20] formulated a weighted multi-objective IoT service placement, optimizing makespan, cost, and energy using population-based meta-heuristic algorithms. The hybrid method GA-SA outperformed other algorithms in their comparative study.

M. Santhosh Kumar and Ganesh Reddy Karri [21] presented a novel technique for scheduling IoT requests in a cloud-fog environment using an adapted ant grey wolf optimization (AGWO). AGWO, combining ant colony optimization and grey wolf optimization, demonstrated improved solution discovery speed and quality.

3. Research Methodology

3.1. Cloud Fog System Model

A three-tiered architecture is comprised of the cloud, fog, and edge layers. At the top tier is the cloud layer, which incorporates cloud servers for the storage and processing of extensive data to address significant challenges associated

with big data. In the ever-expanding realm of IoT, marked by rapid growth, this layer encompasses expansive data centers [2].

The intermediate layer, known as the fog layer, consists of fog machines equipped with specific computing capabilities and mobility support. Serving as a crucial link between the cloud and edge layers, the fog layer provides several advantages, including low-latency processing, data caching, and real-time analytics. Its primary function involves offloading tasks from the cloud layer and distributing them across fog nodes. This layer imparts intelligence to the infrastructure, compensating for the limited compute power of IoT nodes [5]. Fog nodes may possess cognitive abilities, enabling them to monitor and control IoT devices, execute traffic migration in scenarios of concentrated server platform traffic, and maintain a balanced server workload [22].

The bottommost layer, referred to as the IoT devices or edge layer, comprises devices such as laptops, smartphones, cars, PCs, and sensors. Comprising elements that are in close proximity to end-users, this layer is tasked with collecting environmental data and transmitting it to the fog layer for additional processing [20]. Each IoT device is strategically distributed geographically and establishes connections with the Fog node through technologies like Wi-Fi, Bluetooth, and Zigbee [23]. Figure 1 illustrates the architecture of the cloud fog computing system.

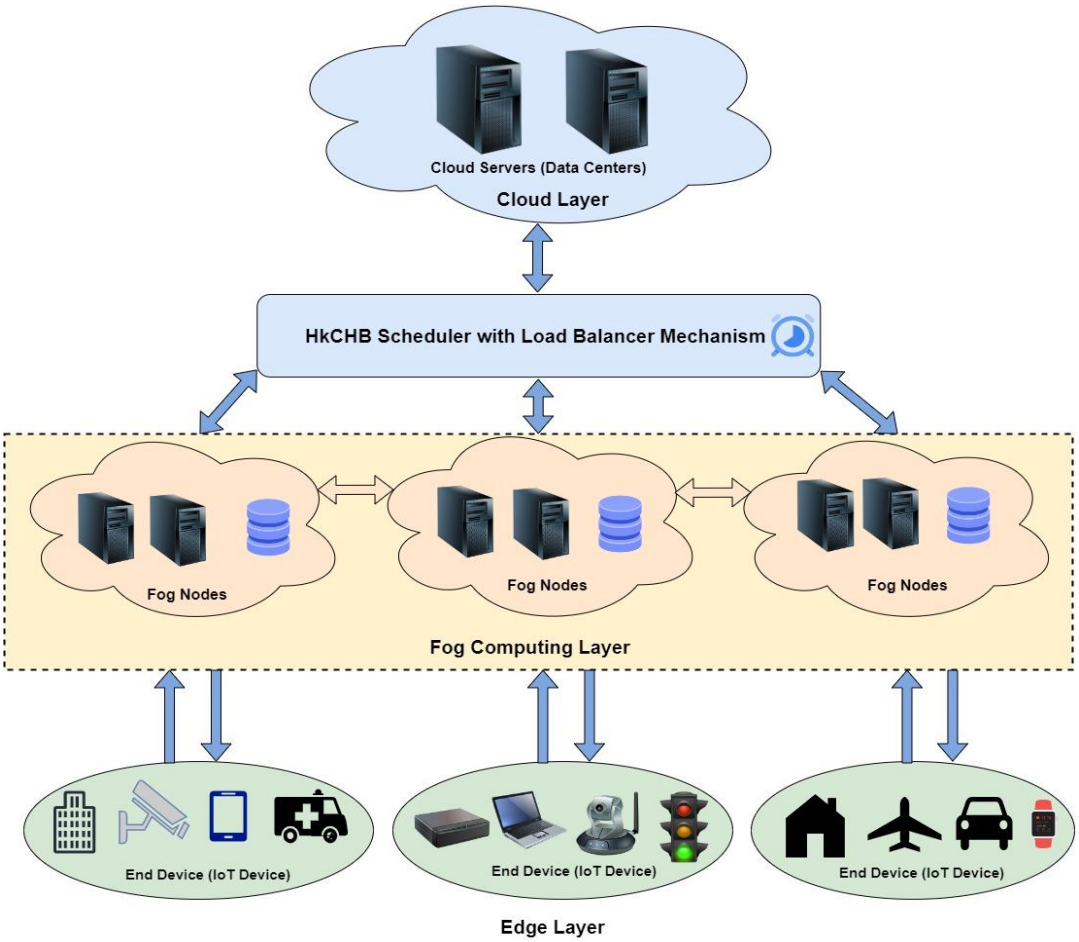

Fig.1. Cloud Fog Computing System Architecture

Task scheduling faces two key challenges: the failure to allocate tasks to virtual machines and the potential allocation of tasks to multiple VMs [15]. Researchers in this field have addressed these issues through the application of load balancing techniques. Figure 2 provides a visual representation of how loads are distributed among VMs with fairness. Users initiate requests directed to Data Centers, and the data center controller subsequently forwards these requests to the Load Balancer. Equipped with load balancing algorithms, the Load Balancer ensures an equitable division of work among VMs, as depicted in figure 2 [24]. These VMs are hosted on physical servers.

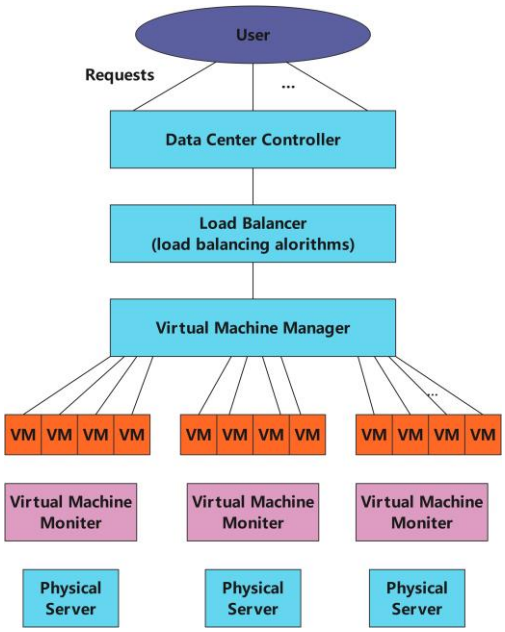

Fig.2. Data Center Load Balancer Mechanism

3.2. Task scheduling Algorithms

Task scheduling in cloud fog systems involves various techniques with load balancing to address the task scheduling problem. This section explores traditional task scheduling algorithms such as Round Robin (RR) and delves into the application of meta-heuristic algorithms, including Ant Colony Optimization (ACO), Honey Bee (HB), and the proposed Hybrid K-means Clustering Honey Bee (HKCHB) algorithm.

3.2.1 Round Robin Algorithm:

The Round Robin scheduler divides time into multiple intervals, assigning each Virtual Machine (VM) a specific time slot. It operates by randomly selecting VMs, which can result in uneven load distribution, some nodes being heavily loaded while others remain lightly loaded. Each VM is allocated a quantum within which it must complete its operations. If a user request finishes within the time quantum, no waiting is required; otherwise, the user must wait for the next slot. Despite its simplicity and ease of implementation, this algorithm has drawbacks, including the need for prior knowledge of user tasks and system resources, and a lack of utilization of the current system state [25, 26, 27].

3.2.2 Ant Colony Optimization Algorithm:

Algorithm 1 Ant Colony Optimization Algorithm (ACO)

```
Input: Number of ant, Iterationmax, attractiveness
τ, and traits η
Output: local best solution, global solution
Initialize the base attractiveness τ, and traits η, for each edge;
for i < Iterationmax do:
  for each ant do:
  choose probabilistically (based on pseudo-random proportional rule) the next
state to move into; add that move to the tabu list for each ant;
  repeat until each ant completed a solution;
  end; // for repeat
  for each ant that completed a solution do:
  update attractiveness τ for each edge that the ant traversed;
 end; // for update
   if (local best solution better than global solution)
     save local best solution as global solution;
   end; // for local and global
  end; // end iteration
```

Fig.3. Pseudo Code of ACO Algorithm.

The ACO meta-heuristic draws inspiration from real ants finding the shortest path between their colonies and a food source. Introduced by Dorigo in 1992, this approach mimics ant behavior by leaving pheromones on the paths they traverse. Pheromone intensity increases with the number of ants using a path and decreases through evaporation. Ants leave more pheromones on the shortest path and fewer on longer paths. Figure 3 presents the pseudo code of the ACO algorithm [28].

3.2.3 Honey Bee Algorithm:

The Honey Bee (HB) algorithm, introduced in 2005, is a population-based search algorithm that emulates the foraging behavior of honey bee colonies. This algorithm combines neighborhood search with global search strategies. In nature, a colony of honey bees efficiently explores numerous food sources in expansive fields, covering distances of up to 11 km. The foraging process begins with scout bees searching for flowers. These scout bees explore flower patches randomly. Upon finding a flower patch, scout bees relay information about the food source's quality, including sugar levels, to their peers waiting in the hive [16]. The scout bees deposit their nectar and then perform a dance on the hive's dance floor to communicate with other bees. The waggle dance, named for the buzzing sound produced by the bees' side-to-side movements, conveys essential information about the discovered food source. The pseudo-code for the Honey Bee algorithm is illustrated in Figure 4 [29].

Algorithm 2 Honey Bee Algorithm (HB)

```
1:  Input: n, m, e, nep, nsp, ngh, MaxIter, and Error
2:  Output: best locations
3:  Generate the initial population size (number of scout bees) as n, set the best
    patch(sites) size as m, set the elite patch size as e, set the number of forager bees
    recruited to the of elite sites as nep, set the number of forager bees around the
    non-elite best patches as nsp, set the neighborhood size as ngh, set the maximum
    iteration number as MaxIter, and set the error limit as Error.
4:  i = 0;
5:  Generate initial population
6:  Evaluate Fitness Value of initial population
7:  Sort the initial population based on the fitness result.
8:  While i<= MaxIter or FitnessValuei , FitnessValuei-1 <= Error
9:      i = i + l;
10:     Select the elite patches and non-elite best patches for neighborhood search
11:     Recruit the forager bees to the elite patches (sites) and non-elite best
        patches.
12:     Evaluate the fitness value of each patch or site.
13:     Sort the results based on their fitness.
14:     Allocate the rest of the bees for global search to the non-best locations.
15:     Evaluate the fitness value of non-best patches.
16:     Sort the overall results based on their fitness.
17:     Run the algorithm until termination criteria met.
18: end // while
```

Fig.4. Pseudo Code of HB Algorithm.

3.2.4 Particle Swarm Optimization Algorithm (PSO):

Particle Swarm Optimization (PSO), established in 1995, has evolved into a mature and widely adopted domain within swarm intelligent algorithms. Its primary objective is to determine the optimal locations of particles for a given function. In 2002, Multi-Objective Particle Swarm Optimization (MOPSO) was introduced, further enhancing the capabilities of PSO. The fundamental premise of PSO lies in the idea that members of a population (swarm) can benefit from both their individual experiences and those of other particles. PSO operates with access to two crucial pieces of information: the best potential solution (PS) and the best PS encountered by its neighboring particles. The algorithm involves three vectors: x, p, and v [30, 31, 32].

1. The x-vector contains the current positional solution (PS) of the particle, with χi representing the fitness assigned to xi by the objective function.
2. The p-vector contains the best PS discovered by a particle, and ρi represents the fitness assigned to the p-vector.
3. The v-vector is instrumental in determining the subsequent PS to be evaluated. Figure 5 illustrates the pseudo-code of the PSO algorithm.

Algorithm 3 Particle Swarm Optimization Algorithm (PSO)

Initialize the swarm population
Initialize *External Archive* (*EA*) (initially *EA* empty)
Set w, c_1, c_2, max num of iterations (g_{max})
Begin
While $g<g_{max}$
For each particle in the swarm
    Select global leaders from Archive
    Select personal best (P_{best})
    Select personal best
    Update velocity according to equation $v_i^{t+1} = wv_i^t + c_1r_1\left(x_{pbest} - X_i^t\right) + c_2r_2\left(x_{gbest} - X_i^t\right)$
    Update position according to equation ($X_i^{t+1} = X_i^t + V_i^{t+1}$)
    Mutation/Perturbation
    Fitness evaluation
    Update P_{best}
end for each
Update External Archive (swarm)
end While
return solutions in Archive

Fig.5. Pseudo Code of PSO Algorithm.

3.2.5 The Proposed Algorithm (Hybrid K-means Clustering Honey Bee Algorithm):

The proposed algorithm, known as the Hybrid K-means Clustering Honey Bee (HKCHB) algorithm, leverages the strengths of both the Honey Bee algorithm and K-means Clustering. K-means is employed as a fitness function to cluster suitable nectar sources (elite patches) into two clusters. Specifically, in addressing the task scheduling with load balancing problem in cloud fog computing, each VM is identified using its suitable VM Id. The clustering method categorizes VMs into two clusters: heavy load VMs and low load VMs. The population size or the number of scout bees is then utilized to select the best elite sites.

In this algorithm, user-requested tasks are allocated to the most suitable VMs in a load-balancing manner until the termination criteria are met. K-Means, a well-known clustering algorithm, is employed to solve the clustering problem. The algorithm classifies objects into a user-defined number of clusters (k clusters). Initially, random cluster centers are chosen, preferably located far from each other. Subsequently, the algorithm calculates the similarity for each point with all cluster centers using Euclidean distance, assigning each point to the most similar cluster [18]. To address the task scheduling with load balancing in cloud computing, the K-means algorithm is utilized as a fitness function for the Honey Bee Algorithm. The VMs are then clustered, and each VM is placed in the most suitable cluster, as depicted in the pseudo-code in Figure 6. Figure 7 illustrates the pseudo-code for the proposed Hybrid K-means Clustering Honey Bee (HCKHB) algorithm.

The distinctive aspect of this algorithm is its ability to address task scheduling and load balancing problems using a hybrid meta-heuristic optimization approach. This innovative method combines K-means and clustering techniques. There are few studies in the literature that solve this problem in this manner.

Algorithm 4 K-means Clustering Algorithm

Input: Vm Id
Output: clustered Vm Id
Assign each object to the closest centroid's cluster.
When all objects have been assigned, recalculate the positions of the centroids (other VMs).
go back to Steps 2 unless the centroids are not changing.

Fig.6. Pseudo Code of K-means Clustering Algorithm

Algorithm 5 Hybrid K-means Clustering Hony Bee (HKCHB)

```
Input: n(Requests), m (VM), Vm Id, e, nep, nsp, ngh, MaxIter, and Error
Output: best sites (Vm Id)
Generate the initial population size (number of scout bees) as n, set the best
patch(sites) size as m, set the elite patch size as e, set the number of forager bees
recruited to the of elite sites as nep, set the number of forager bees around the
non-elite best patches as nsp, set the neighborhood size as ngh, set the maximum
iteration number as MaxIter, and set the error limit as Error.
i = 0;
Generate initial population (Number of requested tasks)
Evaluate Fitness Value of initial population
Sort the initial population based on the fitness result.
While i<= MaxIter or FitnessValuei, FitnessValuei-1 <= Error
    i = i + l;
Select the elite patches (Suitable VM Id).
Recruit the forager bees (Number of task) to the elite patches (Suitable VM Id)
Evaluate the fitness value of each patch (VM) using K-means.
Sort the results based on their fitness.
Allocate the rest of the bees for global search to the non-best locations (Rest
task to VMs).
Evaluate the fitness value of non-best patches (Evaluate non suitable VM Id
using K-means).
Sort the overall results based on their fitness.
Run the algorithm until termination criteria met.
end // while
```

Fig.7. Pseudo Code for Hybrid K-means Clustering Honey Bee Algorithm

4. Experiments and Results

4.1. Experiments:

This research conducts a comparative analysis of four scheduling algorithms with load balancing: RR, ACO, HB, and HKCHB. The aim is to generate optimal solutions for NP-hard problems within polynomial time [23]. Various simulators are available to simulate cloud fog systems for solving task scheduling problems, including CloudSim, CloudAnalyst, CloudReports, CloudExp, GreenCloud, iCanCloud, and iFogSim [33]. For this research, the CloudSim simulator is employed, with the integration of the proposed algorithm. CloudSim, developed at the GRIDS laboratory at the University of Melbourne, provides graphical output presentations and allows users to simulate applications across multiple data centers and user groups. The simulation involves adding Java code for the proposed algorithm to CloudSim as a novel task scheduling and load balancing technique. The Java code with a graphical user interface (GUI) for CloudSim is obtained from GitHub [34]. Simulator parameters, including user configuration, data center configuration, and VM configuration, are defined.

Two scenarios are employed in the experiments: one with a single data center (DC1) and another with two data centers (DC2). The simulation analysis parameters, including CPU capacity, are listed in Table 1. The simulations are conducted on a system equipped with an Intel Core i5-1135G7 processor, running at 2.4 GHz, and 8 GB of RAM.

Table 1. Cloud-Fog computing parameters Using CloudSim

Parameter	Value
Number of cloud data centre	1, and 2
Number of cloud data VM's	5, 10, 25, 50, and 100
Number of cloud data VM's	10, 15, 30, 60, and 150
Number of end device	10
Computing power (MIPS) for cloud	[3000:5000]
Computing power (MIPS) for fog	[1000:2000]
Number of tasks	500, 1000, 1500, 2000, 2500, and 3000

The experiments were conducted in a heterogeneous environment featuring hosts with varying numbers of CPUs and speeds. This diversity allows for the measurement of response (execution) time, processing time, and throughput.

Figures 8 and 9 explicitly consider the virtual machines (VMs) within this heterogeneous host environment. Each machine is characterized by distinct CPU counts and processing speeds, accentuating the realistic and varied conditions under which the experiments were conducted.

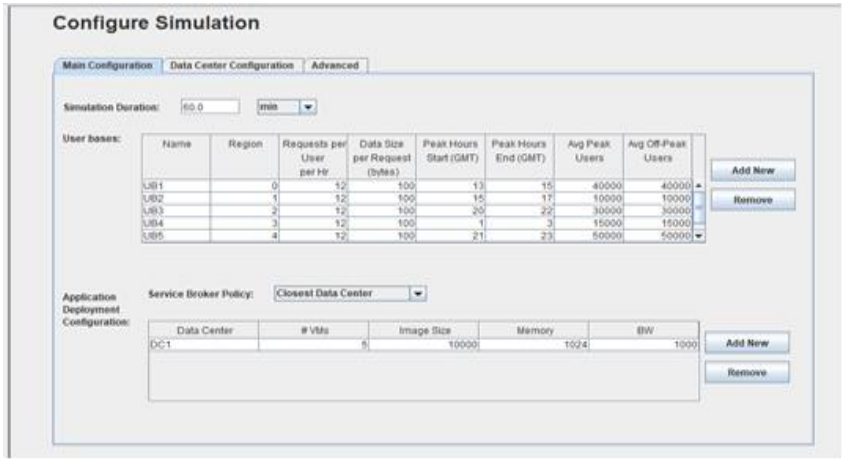

Fig. 8. User and VMs Configuration

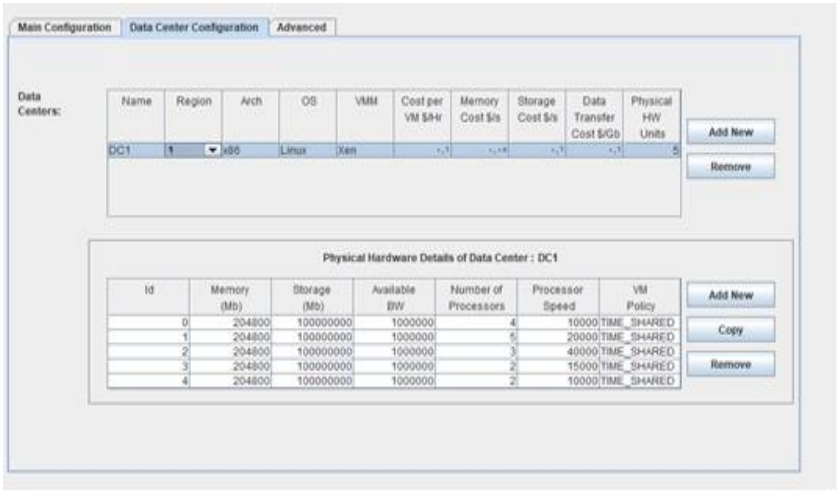

Fig.9. Data Centers Configuration

4.2. Results and Discussion:

Upon completing the simulations to measure response time, processing time, and throughput, the results are summarized as follows:

Figure 10 illustrates the superior performance of the proposed HKCHB algorithm in reducing overall execution time, particularly evident in scenarios involving multiple tasks within a heterogeneous environment. The HKCHB scheduler optimizes resource allocation through a hybrid approach integrating K-means and honeybee algorithms. This ensures that all tasks are allocated to optimal virtual machines (VMs) in a load-balanced manner, minimizing time wastage.

Comparing the proposed algorithm with RR, ACO, HB, and PSO, it becomes apparent that as the number of tasks increases, the time required for these algorithms to complete experiences exponential growth. The HKCHB algorithm consistently demonstrates the lowest execution time when benchmarked against PSO, HB, ACO, and RR. Notably, it achieves a reduction in execution time by 11.5% compared to PSO, 16.8% compared to HB, 18.5% compared to ACO, and a substantial 41.7% compared to RR.

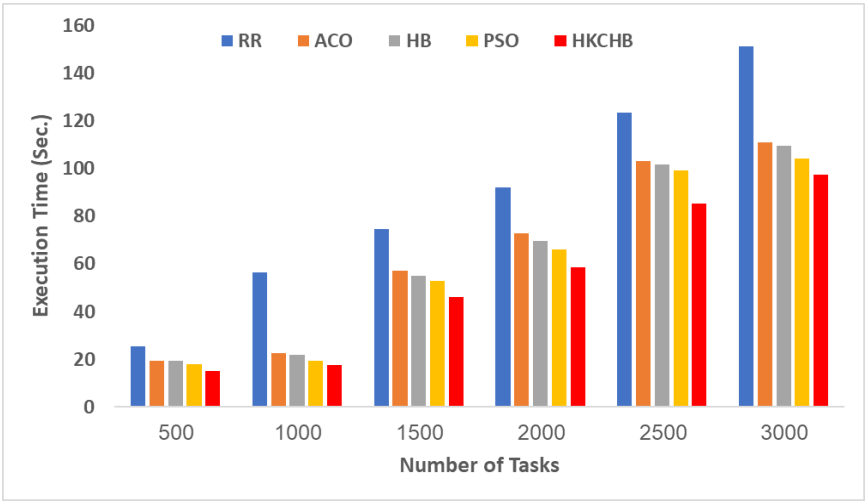

Fig.10. Execution Time Vs Number of Tasks

The HKCHB algorithm, as proposed, stands out as a highly efficient solution, consistently achieving the lowest processing time when compared to established algorithms such as PSO, HB, ACO, and RR. This superiority is vividly depicted in Figure 11, which presents the optimal processing time results for real-time workloads with varying task sizes.

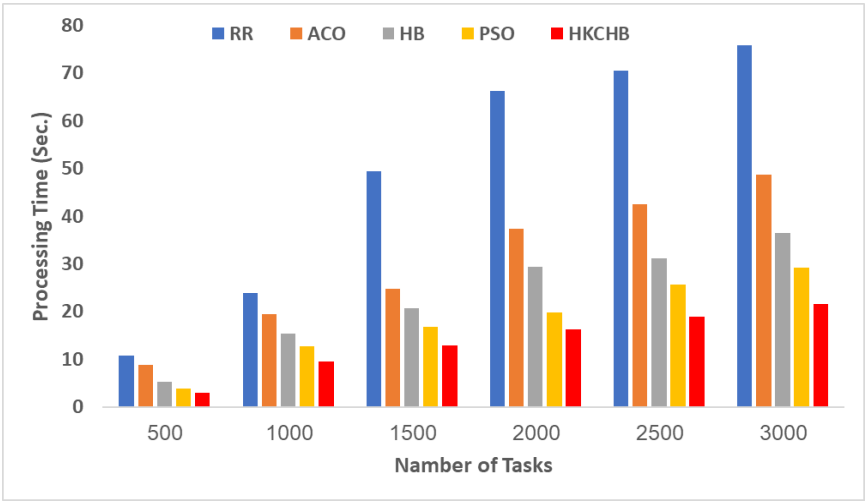

Fig.11. Processing Time Vs Number of Tasks

The graph underscores that HKCHB is not only effective but also an efficient approach in addressing critical challenges associated with cloud fog task scheduling in a heterogeneous environment. Specifically, it demonstrates a substantial reduction in processing time compared to other algorithms, showcasing its efficiency. The HKCHB algorithm achieves remarkable processing time reductions, including 23.7% compared to PSO, 41% compared to HB, 55.7% compared to ACO, and an impressive 71.2% compared to RR. This emphasizes the algorithm's effectiveness in optimizing processing time for real-time workloads, contributing to the overall efficiency of cloud fog computing systems.

In the depicted Figure 12, a comparison is made regarding the throughput achieved by RR, HB, ACO, PSO, and the proposed HKCHB algorithm. The horizontal axis represents the number of tasks, while the vertical axis indicates throughput. These results unequivocally establish the superior performance of HKCHB compared to other algorithmic approaches in terms of throughput within a heterogeneous environment. Specifically, HKCHB demonstrates a notable increase in throughput, surpassing other algorithms by 9.7% compared to PSO, 24.7% compared to HB, an impressive 78.6% compared to ACO, and a remarkable 268.8% compared to RR. This signifies that, upon implementing the HKCHB algorithm in the task scheduling and load balancing processes of a cloud fog computing system, the system can efficiently process tasks within a given timeframe. This heightened efficiency is anticipated to significantly enhance the overall performance of cloud fog computing systems in the foreseeable future.

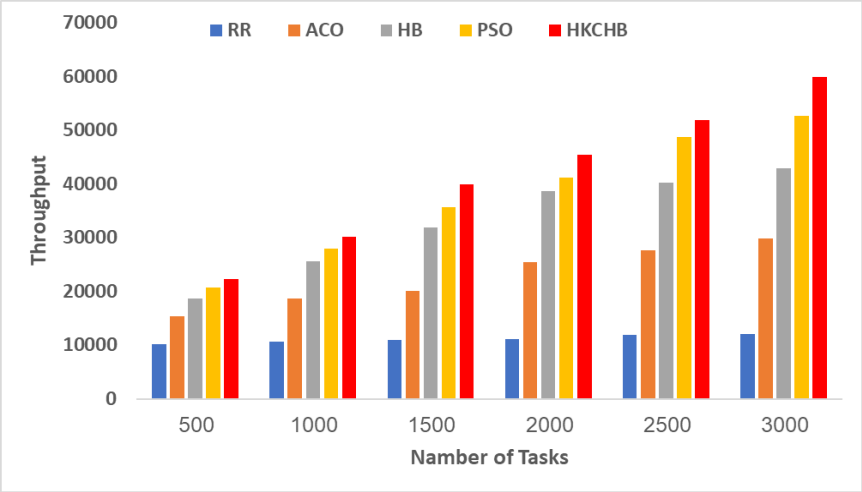

Fig. 12. Throughput Vs Number of Tasks

Table 2 and Figure 13 provide a visual representation of the improved percentages achieved by the HKCHB algorithm compared to other algorithms. The HKCHB algorithm not only enhances throughput when compared to all other algorithms but also achieves reductions in both execution time and processing time, showcasing its superiority. The study establishes the effectiveness and robustness of this proposed algorithm through these measurements. Nonetheless, future investigations should aim to further validate the algorithm's strength by incorporating additional measurements if feasible.

Execution Time Reduction: Response time, also known as execution time, indicates the total duration required for a computer to complete a task. This encompasses various factors such as disk accesses, memory operations, I/O activities, operating system overhead, and CPU execution time. The HKCHB algorithm demonstrates reduced execution time, with reductions of 11.5% compared to PSO, 16.8% compared to HB, 18.5% compared to ACO, and 41.7% compared to RR. This suggests that by implementing the proposed algorithm, along with the increasing volume of data (Big Data) in the realm of cloud fog computing and the Internet of Things (IoT), there will be a simpler means to adjust scalability in real-time systems.

Throughput Enhancement: On the other hand, Throughput serves as a metric that quantifies the number of information units a system can process within a specified timeframe. This metric finds broad applications across diverse systems, spanning computer and network systems to organizational contexts [21]. HKCHB demonstrates an enhanced throughput compared to other algorithms, with increases of 9.7% over PSO, 24.7% over HB, 78.6% over ACO, and a remarkable 268.8% over RR. This makes the proposed algorithm potentially more productive (in terms of Throughput) and capable of processing information within real-time systems. This is crucial for any system relying on cloud fog computing.

Processing Time Reduction: In contrast, CPU time (or process time) specifically refers to the duration during which a central processing unit (CPU) is actively engaged in processing instructions of a computer program or operating system. This excludes elapsed time, which accounts for waiting periods for input/output (I/O) operations or transitions to low-power (idle) mode [35]. HKCHB demonstrates decreased processing time, with reductions of 23.7% compared to PSO, 41.0% compared to HB, 55.7% compared to ACO, and a substantial 71.2% compared to RR. According to the results of the proposed algorithm, processing time was reduced, which will inevitably lead to accelerating information processing and reducing delays. This is particularly important in real-time systems, especially in fog cloud computing networks.

Table 2. Enhanced Percentages of HKCHB Algorithm over Other Algorithms

Metric	HKCHB vs PSO	HKCHB vs HB	HKCHB vs ACO	HKCHB vs RR	Overall Percent
Throughput (%)	9.7%	24.7%	78.6%	268.8%	**95.4%**
Execution Time	11.5%	16.8%	18.5%	41.7%	**22.1%**
Processing Time	23.7%	41.0%	55.7%	71.2%	**47.9%**

In summary, the presented results showcase a significant enhancement in execution time, processing time, and throughput across various synthetic workloads. The conclusive finding is that the devised HKCHB algorithm consistently outperforms other methods, affirming its effectiveness and efficiency in addressing optimization challenges within cloud-fog task scheduling. On average, the HKCHB algorithm stands out as the most successful, underscoring its effectiveness in handling complex cloud-fog task scheduling optimization problems.

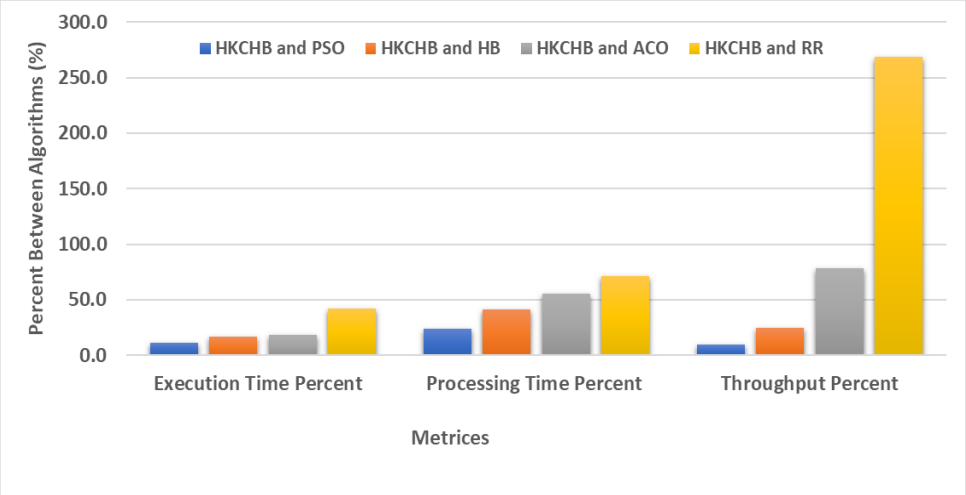

Fig.13. Enhanced Percentages of HKCHB Algorithm over Other Algorithms

The conclusive findings indicate that the developed HKCHB algorithm is not only effective but also efficient in tackling the challenges of cloud-fog task scheduling optimization. Its superior performance across multiple metrics positions it as a promising solution for enhancing the overall efficiency of cloud-fog computing systems.

Limitations

While the HKCHB algorithm offers several advantages for task scheduling and load balancing in cloud fog computing, it also comes with certain limitations that need to be taken into account. Here are some of the limitations:

- Parameter Setting: Achieving optimal performance with the HKCHB algorithm requires accurate parameter settings, similar to other optimization algorithms. Incorrect parameter values can significantly impact the results, necessitating careful adjustment for optimal performance.
- Increase in Data Volume: The exponential growth of data generated by cloud fog computing environments, particularly due to advancements in Internet of Things (IoT) technology, can affect scalability. This includes challenges such as increased search space size, computational complexity, and memory requirements. Large-scale task scheduling problems may pose difficulties in terms of computational time and resource consumption.
- Linking Research Results to Reality: The applicability of research results obtained through simulation to real-world scenarios, particularly in manufacturing, may vary. It is essential to validate these results on the ground to ensure their practical efficacy.

To effectively address these real-world challenges and tackle complex problems such as task scheduling and load balancing in cloud fog computing (which are NP-hard problems), the development of optimization algorithms like HKCHB was necessary. To enhance the robustness of this algorithm, hybridization techniques, such as incorporating clustering processes and utilizing K-means, were introduced. These efforts were centered around making the algorithm more practical and realistic to improve its performance and scalability.

5. Conclusion and Future Work

Cloud fog computing is an indispensable on-demand service, yet the optimization of task scheduling with load balancing has emerged as a critical concern, denoted as an NP-hard problem. Numerous researchers are actively engaged in addressing this challenge. Based on the fundamental research idea, this study proposes the use of optimization algorithms inspired by nature to enhance the efficiency and scalability of distributed systems. Whereas, in this research paper a novel task scheduling and load balancing algorithm named HKCHB, relying on a hybrid meta-heuristics approach that combines K-means clustering with the Honey Bee optimization algorithm was introduced. Meta-heuristics are instrumental in providing solutions that are not only reasonable but also characterized by reduced complexity. The empirical results affirm the superiority of the proposed HKCHB algorithm over established counterparts such as PSO, HB, ACO, and RR. The evaluation is conducted across various scenarios involving different numbers of virtual machines (VMs) within a single data center (DC) or two DCs in a heterogeneous environment. The HKCHB algorithm demonstrates enhanced performance in terms of response (execution) time, processing time, and throughput. Notably, it achieves a substantial reduction in execution time by 22.1% and processing time by 47.9% compared to other algorithms. Moreover, it exhibits a remarkable increase in throughput, surpassing other algorithms by 95.4%. The proposed work suggests future directions aimed at refining new algorithms tailored for cloud fog computing

systems. This endeavor holds the promise of not only advancing the efficiency of cloud fog computing but also contributing to the optimization of Internet of Things (IoT) systems.

References

[1] Tran Cong Hung, and Nguyen Xuan Phi, "STUDY THE EFFECT OF PARAMETERS TO LOAD BALANCING IN CLOUD COMPUTING," International Journal of Computer Networks & Communications (IJCNC), Vol.8, No.3, May 2016.

[2] Kunal S, Saha A, Amin R, "An overview of cloud-fog computing: Architectures, applications with security challenges," Security and Privacy, 2019. https://doi.org/10.1002/spy2.72.

[3] Sabireen H., Neelanarayanan V., "A Review on Fog Computing: Architecture, Fog with IoT," Algorithms and Research Challenges, ICT Express, Volume 7, Issue 2, 2021, https://doi.org/10.1016/j.icte.2021.05.004.

[4] Mohamed Abd Elaziz, Laith Abualigah, Ibrahim Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," Future Generation Computer Systems, Volume 124, 2021, https://doi.org/10.1016/j.future.2021.05.026.

[5] M. Santhosh Kumar, K. Ganesh Reddy, Rakesh Kumar Donthi, "SSKHOA: Hybrid Metaheuristic Algorithm for Resource Aware Task Scheduling in Cloud-fog Computing," International Journal of Information Technology and Computer Science, Vol.16, No.1, pp.1-12, 2024.

[6] Gurpreet Singh Bedi, "An Efficient Load Balancing based on Resource Utilization in Cloud Computing," Master of Engineering in Software Engineering, 147004, June 2014.

[7] Hafiz Jabr Younis, Alaa Al Halees, and Mohammed Radi, "Hybrid Load Balancing Algorithm in Heterogeneous Cloud Environment," International Journal of Soft Computing and Engineering (IJSCE), Volume-5 Issue-3, July 2015.

[8] Jafar Shayan, Ahmad Azarnik, Suriayati Chuprat, Sasan Karamizadeh, and Mojtaba Alizadeh, "Identifying Benefits and Risks Associated with Utilizing Cloud Computing," The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, 2013.

[9] Ali Al-maamari, and Fatma A. Omara, "Task Scheduling Using PSO Algorithm in Cloud Computing Environments," International Journal of Grid Distribution Computing Vol. 8, No.5, (2015).

[10] Mala Kalra, and Sarbject Singh, "A review of metaheuristic scheduling techniques in cloud computing," Egyptian Information Journal, August (2015).

[11] Ashima, and Mrs Navjot Jyoti, "Enhancing Job Allocation Using NBST in Cloud Environment: A Review," International journal of computer and technology, June 2017.

[12] Kousik Dasgupta, Brototi Mandal, Paramartha Dutta, and Jyotsna Kumar Mondal, Santanu Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing," Elsevier, International Conference on Computational Intelligence: Modelling Techniques and Applications (CIMTA) 2013.

[13] Safwat A. Hamad, and Fatma A. Omara, "Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment," International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 7, No. 4, 2016.

[14] Gao, R.; Wu, J. "Dynamic Load Balancing Strategy for Cloud Computing with Ant Colony Optimization," Future Internet 2015, 7, 465-483. https://doi.org/10.3390/fi7040465.

[15] A.I.Awad, N.A.El-Hefnawy, and H.M.Abdel_kader, "Enhanced Particle Swarm Optimization For Task Scheduling In Cloud Computing Environments," Elsevier, International Conference on Communication, Management and Information Technology (ICCMIT), 2015.

[16] Baris Yuce, Michael S. Packianather, Ernesto Mastrocinque, Duc Truong Pham, and Alfredo Lambiase, "Honey Bees Inspired Optimization Method: The Bees Algorithm," mdpi Journal, Insects November 2013.

[17] Duc Truong, and Marco Castellani, "A comparative study of the Bee Algorithm as tool for function optimization," Pham & Castellani, Cogent Engineering, 2015.

[18] Bashar Al-Shboul, and Sung-Hyon Myaeng, "Initializing K-Means using Genetic Algorithms," Conference Paper, November 2009.

[19] F. A. Saif, R. Latip, Z. M. Hanapi and K. Shafinah, "Multi-Objective Grey Wolf Optimizer Algorithm for Task Scheduling in Cloud-Fog Computing," in IEEE Access, vol. 11, pp. 20635-20646, 2023m doi: 10.1109/ACCESS.2023.3241240.

[20] Hemant Kumar Apat, Bibhudutta Sahoo, Veena Goswami, Rabindra K. Barik, "A hybrid meta-heuristic algorithm for multi-objective IoT service placement in fog computing environments," Decision Analytics Journal, Volume 10, 2024, https://doi.org/10.1016/j.dajour.2023.100379.

[21] M. Santhosh Kumar & Ganesh Reddy Karri, "AGWO: Cost Aware Task Scheduling in Cloud Fog Environment Using Hybrid Metaheuristic Algorithm," International Journal of Experimental Research and Review. 33, 41-56, 2023, DOI: https://doi.org/10.52756/ ijerr. 2023.v33spl.005.

[22] Adam A. Alli, Muhammad Mahbub Alam, "The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications," Internet of Things, Volume 9, 2020, https://doi.org/10.1016/j.iot.2020.100177.

[23] Jamil B, Shojafar M, Ahmed I, Ullah A, Munir K, Ijaz H, "A job scheduling algorithm for delay and performance optimization in fog computing Concurrency," Computat Pract Exper, 2019; e5581. https://doi.org/10.1002/cpe.5581.

[24] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader, "Load-balancing algorithms in cloud computing: A survey," ELSEVIER, Journal of Network and Computer Applications, Volume 88, 15 June 2017, Pages 50-71.

[25] Deepika, Divya Wadhwa, and Nitin Kumar, "Performance Analysis of Load Balancing Algorithms in Distributed System," Advance in Electronic and Electric Engineering. ISSN 2231-1297, Volume 4, Number 1, 2014.

[26] Reena Panwar, Bhawna Mallick, "A Comparative Study of Load Balancing Algorithms in Cloud Computing," International Journal of Computer Applications (0975 – 8887) Volume 117 – No. 24, May 2015.

[27] Sandip Patel, Ritesh Patel, Hetal Patel, and Seema Vahora, "CloudAnalyst: A Survey of Load Balancing Policies," International Journal of Computer Applications (0975 – 8887) Volume 117 – No. 21, May 2015.

[28] Abhijit Patil, Harshal Gala, and Jai Kapoor, "Dynamic Load Balancing in Cloud Computing using Swarm Intelligence Algorithms," International Journal of Computer Applications (0975 – 8887) Volume 130 – No.15, November 2015.

[29] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "Bee Algorithm A Novel Approach to Function Optimization," Technical Note: MEC 0501, December 2005.
[30] Moore, Jacqueline and Richard Chapman, "Application of Particle Swarm to Multiobjective Optimization." Computer Systems: Science & Engineering, 1999.
[31] C. A. Coello Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), Honolulu, HI, USA, 2002, pp. 1051-1056 vol.2, doi: 10.1109/CEC.2002.1004388.
[32] Mahmoud Moshref, Rizik Al-Sayyed, Saleh Al-Sharaeh, "MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS FOR WIRELESS SENSOR NETWORKS: A COMPREHENSIVE SURVEY," Journal of Theoretical and Applied Information Technology, Vol.98. No 14, July 2020.
[33] Khadijah Bahwaireth, Loai Tawalbeh, Elhadj Benkhlifa, and Yaser Jararweh, "Experimental Comparison of Simulation Tools for Efficient Cloud and Mobile Cloud Computing Application," EURASIP Journal on Information Security, June 2016.
[34] https://github.com/suhailgupta03/Cloud_Analyst_In_Progress.
[35] Mohamed Abdel-Basset, Doaa El-shahat, Mohamed Elhoseny, Houbing Song, "Energy-Aware Metaheuristic Algorithm for Industrial Internet of Things Task Scheduling Problems in Fog Computing Applications," IEEE Internet of Things Journal, August 2020.

Authors' Profiles

Mahmoud Moshref holds a bachelor's degree in computer science from the An-Najah University of Nablus, Palestine in 2003. His master's degree in computing was obtained at the Birzeit University of Birzeit, Ramallah, Palestine, in 2012 and was focused on the improving RFID system for Internet of Things. He received the Ph.D. degree in Computer Science from The University of Jordan, Amman, Jordan in 2021, and was focused on using Multi-objective Metaheuristic algorisms to improving QoS in Wireless Sensor Network, he is currently an Assistant Professor in the School of Information Technology, Computer Science Department, Palestine Technical University, Tulkarem, Palestine, he interested in networks, network security, data mining, Machine Learning, cloud computing, fog Computing, and internet of things.

Sherin Hijazi Sherin Hijazi earned her bachelor's degree in management information systems from An-Najah National University in Nablus, Palestine, in 2005. She then pursued a master's degree in computer information systems at Al-Yarmouk University in Irbid, Jordan, completing it in 2012. In 2015, she received a scholarship from Palestine Technical University-Kadoorie (PTUK) to pursue her Ph.D. in Computer Science at the University of Jordan in Amman, Jordan, which she successfully obtained in 2020.

From 2005 to 2007, Sherin served as an Administrative Assistant in the IT Department at Palestine Securities Exchange (PSE). From 2007 to 2011, she worked as a Programmer at PTUK in Tulkarm, Palestine. She then progressed to leadership roles, serving as Head of Programming at the Computer Center at PTUK from 2011 to 2013. Following this, she contributed to academia as a Lecturer in the Department of Applied Computing at PTUK from 2013 to 2020. Continuing her academic career, she became an Assistant Professor in the Department of Applied Computing from 2020 to 2022. In 2022, she transitioned to the role of Assistant Professor in the Department of Computer Science and simultaneously served as the Head of the Computer Science Department and the Head of the Information System Department within the Information Technology Faculty at PTUK from 2022 to 2023. Since 2022 and until now, she has been leading the Computer Science Department within the Information Technology Faculty at PTUK.

Dr. Hijazi has made significant contributions to the field with nine publications covering various areas of computer science. Her research interests include artificial intelligence, data science, data analysis, algorithms, network security, software engineering, and database systems. In recognition of her outstanding contributions, Dr. Hijazi received the IEEE Systems Journal Best Paper Award in 2020. Additionally, she was honored with the Reviewer Certificate from the Security and Privacy Journal for the period 2020-2024 and the Reviewer Certificate from IEEE Access Journal for the period 2020-2022.

Azzam Sleit holds B.Sc, M.Sc. and Ph.D. in Computer Science. He received his Ph.D. in 1995 from Wayne State University, Michigan. Azzam Sleit is the Former Minister of Information and Communications Technology (2013-2015). He was functioning as the Dean of King AbdullaII School for Information Technology and a Professor of Computer Science with the University of Jordan, where he had also functioned as the Assistant President and Director of the Computer Center (2007-2009). Before joining the University of Jordan in 2005, Prof. Sleit was the Chief Information Officer at Hamad Medical/ Ministry of Public Health, Qatar. Prof. Sleit has over twenty-five years of experience and leadership working at all levels of government, private and international sectors. Prof. Sleit has participated in numerous research activities related to Cloud Computing, Imaging Databases, Data Mining, Health and Management Information Systems and Software Engineering. He authored more than sixty refereed research papers published in reputable journals and conferences. Since 1987, Prof. Sleit has taught Computer Science courses at various universities in the United States and Middle East maintaining high teaching standards.

Ahmad Sharieh holds Doctor of Philosophy in Computer and Information Sciences, BSc in Mathematics, BSc in Computer Science, MS in Computer Science, and High Diploma in Higher Education Head of Faculty at University of Jordan, Prof. Sharieh was used to be dean of King Abdullah School for Information Technology at The University of Jordan. Prof. Sharieh published several articles in Journals, conferences, and authored and prepared 14 books. He gained grant for eight research projects from UJ and EUROPE. He developed Several software systems such as: Teaching Sign Language, E-learning Modelling and Simulation, and Online (Automated) Exams. He is on the editorial board of several journals and conferences, and a referee of several others. He has supervised a number of Master and PH.D. research students. Prof. Sharieh has participated in numerous research activities related to Distributing Systems, Expert Systems, E-Government, E-Learning, Parallel Processing, Pattern Recognition, Software Engineering, Wire/Wireless Communication, Modeling and Simulation, and Cloud Computing.

How to cite this paper: Mahmoud Moshref, Sherin Hijazi, Azzam Sleit, Ahmad Sharieh, "HKCHB: Meta-heuristic Algorithm for Task Scheduling and Load Balancing in Cloud-fog Computing", International Journal of Engineering and Manufacturing (IJEM), Vol.14, No.3, pp. 1-15, 2024. DOI:10.5815/ijem.2024.03.01