Article

Multi-Objective Meta-Heuristic Dynamic Load Balancing and Resource Allocation Approach in Cloud Computing Using Harris Hawk Optimization Algorithm (MDLB-HHO)

Paulraj Dasan , Sethukarasi Thirumaaran , Vigilson Prem Monickaraj [*] , Neelakandan Subramani

Posted Date: 25 July 2023

doi: 10.20944/preprints202307.1612.v1

Keywords: load balancing; job scheduling; cloud computing; harris hawk optimization

Article

Multi-Objective Meta-Heuristic Dynamic Load Balancing and Resource Allocation Approach in Cloud Computing Using Harris Hawk Optimization Algorithm (MDLB-HHO)

Paulraj Dasan [1], Sethukarasi Thirumaaran [1], Vigilson Prem Monickaraj [2,*] and Neelakandan Subramani [1]

[1] Department of Computer Science and Engineering, R.M.K. Engineering College, Chennai, 601206, India; kingrajpaul@gmail.com, tsk.cse@rmkec.ac.in, snksnk17@gmail.com
[2] Department of Computer Science and Engineering, R.M.K. College of Engineering and Technology, Chennai, 601206, India; vigiprem@gmail.com
* Correspondance: vigiprem@gmail.com

Abstract: The ultimate aim of dynamic load balancing in cloud computing systems is to maximise the efficiency with which resources are utilised and workloads are distributed. Given that load balancing is a multi-objective process and that response time is a priority, the Harris hawk optimisation (HHO) algorithm was developed as a unique solution for dynamic load balancing. Based on burden distribution and resource utilisation, the HHO algorithm is responsible for dynamically assigning workloads to virtual machines (VMs). Through iterative interactions and position updates, the hawks investigate the solution space, determine the optimal method for dividing the work, and adapt to the ever-changing conditions of the workload. The HHO algorithm has been demonstrated to be effective and efficient in the management of dynamic load balancing via a series of experimental evaluations and comparisons with other load-balancing approaches. These discoveries have led to quicker response times and more efficient resource utilisation. Utilising the collaborative search behaviour of hawks, this technique provides a solution that is both practicable and effective for addressing load balancing concerns in dynamic scenarios.

Keywords: load balancing; job scheduling; cloud computing; harris hawk optimization

1. Introduction

Cloud computing is a paradigm-shifting technology that gives users access to the resources of computer systems in a scalable and on-demand manner. Load balancing and efficient resource allocation become more important as the number of cloud-based services and applications grows. These are two of the most important factors in optimising performance and making the most of the resources that are available. Techniques for load balancing allow for the dynamic distribution of tasks over several virtual machines, which in turn improves response times and resource utilisation. In the past, load balancing solutions concentrated on either minimising the amount of time needed for reactions or increasing the amount of resources that were employed. On the other hand, when it comes to cloud computing in the real world, load balancing is a multi-objective problem that necessitates making compromises between competing goals. For instance, optimising resource utilisation may result in longer reaction times, whilst lowering response times may lead to a more efficient use of available resources. Because of this, there is a requirement for load balancing strategies that are capable of properly dealing with the multi-objective character of the problem.

By modelling natural occurrences and biological processes, meta-heuristic algorithms have shown exceptional success in finding solutions to difficult optimisation challenges. The HHO (Harris hawk optimisation) algorithm is one of such type of algorithms. It gets its name from the cooperative hunting behaviour of Harris's hawks. Due to HHO's capacity to find a balance between exploration and exploitation, it has shown great potential in a variety of optimisation domains, which makes it a good candidate for addressing the multi-objective load balancing and resource allocation problem in cloud computing. HHO has shown tremendous potential in numerous optimisation domains.

Using the HHO algorithm, this work proposes a unique solution for multi-objective dynamic load balancing and resource allocation in cloud computing. Our objective is to achieve an effective and well-balanced distribution of activities and resources inside cloud settings by simultaneously optimising a number of objectives, such as reaction time, resource utilisation, energy efficiency, and cost. We provide a comprehensive and effective solution by incorporating the multi-objective aspect of the problem into the HHO algorithm.

The HHO algorithm is utilised by the proposed approach in order to dynamically distribute jobs across the available virtual machines (VMs) and assign resources in accordance with the workload and resource utilisation [1]. The approach takes advantage of the hierarchical split of hawks into several functions, such as exploratory hawks, sentinel hawks, and possessive hawks, in order to evaluate the cooperative behaviour that is seen in Harris's hawks while they are hunting. These responsibilities make it possible for the hawks to investigate the solution space, keep an eye on the quality of the proposed solutions, and work towards the most effective decisions about load balancing and resource distribution.

In order to determine how effectively the proposed approach performs, we conduct quite a lot of experiments and make comparisons with several load balancing and resource allocation approaches that are existing in cloud computing. In order to evaluate the performance, we examine important performance parameters such as reaction time, resource utilisation, energy usage, and cost. The findings point to the superiority of a multi-objective meta-heuristic dynamic load balancing and resource allocation strategy that makes use of the HHO algorithm. This technique demonstrates its capacity to accomplish better trade-offs between conflicting objectives and increase overall system performance.

The remainder of the paper is organised as follows: Section 2 provides a comprehensive literature review on cloud computing load balancing, resource allocation, and meta-heuristic algorithms. Section 3 presents the methodology, including problem formulation, HHO algorithm details, and multi-objective optimisation integration. Section 4 describes the experimental setup, presents the results, and provides a comprehensive analysis. Section 5 concludes by summarising the findings, discussing the contributions of this research, and outlining prospective directions for further improvement and research in multi-objective dynamic load balancing and resource allocation using the HHO algorithm in cloud computing.

2. Literature Review

In cloud computing systems, load balancing and resource allocation are essential components because they have the potential to enhance system performance, boost resource utilisation, and guarantee that resources are distributed fairly among users. Techniques for load balancing and resource allocation in cloud computing systems that are inspired by natural events or biological behaviours show a lot of promise. These algorithms are called meta-heuristic algorithms. The Harris hawk optimisation (HHO) method, which takes its name from the hunting strategy of Harris hawks, has been shown to be effective in the resolution of a variety of optimisation issues. However, its application in the process of load balancing and resource allocation in cloud computing is yet substantially unexplored.

This literature review focuses on meta-heuristic methods, such as genetic algorithms, particle swarm optimisation, ant colony optimisation, and simulated annealing, for load balancing and resource allocation within the context of cloud computing. It examines their advantages and disadvantages, as well as their load balancing and resource distribution applications. D. A. Shafiq and coworkers provide a load-balancing mechanism for data centres to ensure consistent application performance in cloud computing. In order to accomplish load balancing in the data centres, the load-balancing algorithm that they proposed makes use of a multi-objective optimisation strategy. The efficiency of the algorithm is further confirmed using sensitivity analysis, which evaluates the influence that changing parameters has on the performance of the algorithm. The authors offer their perspectives on the resilience of the algorithm as well as its capacity to adjust to a variety of different workload circumstances. A unique technique that handles the issues of load distribution and resource allocation is presented in this study as a contribution to the field of load balancing in cloud computing. It provides helpful insights as well as practical consequences for creating load-balancing algorithms that are efficient in cloud-based contexts.

The paper [4] analyses the significance of load balancing in cloud computing and draws attention to the restrictions imposed by the currently available load balancing solutions when it comes to the management of load distribution in big-data cloud systems. The authors propose a unique load balancing strategy that they term the Central-Regional Architecture Based Load Balancing Technique (CRLBT). This strategy is intended to help address the problem. Combining a specified throughput maximisation technique with Throughput Maximised-Particle Swarm Optimisation (TM-PSO) and Throughput Maximised-Firefly optimisation (TM-Firefly) algorithms gives CRLBT a distinct advantage over typical central, distributive, and hierarchical cloud systems. In addition, CRLBT distinguishes itself from these cloud designs by combining these algorithms. The results of the experiments demonstrate considerable improvements in response time, task rejection ratio, CPU utilisation rate, and network throughput, which confirms the efficiency of the suggested technique in providing superior load balancing within the context of big-data cloud systems.

The paper [5] proposes a method, DEER, for load balancing in fog computing environments, taking into account the increasing significance of fog computing in managing the information flow in large and complex networks for the Internet of Things (IoT). The objective is to increase overall efficiency while decreasing energy consumption, carbon emissions, and energy costs.

The most important components of the DEER strategy are the Tasks Manager, Resource Information Provider, Resource Scheduler, Resource Load Manager, Resource Power Manager, and Resource Engine. Tasks are submitted via Tasks Manager, and Cloud Data Centres register resource information. The Resource Engine is responsible for allocating work, while the Resource Scheduler organises available resources based on their utilisation. While Resource Power Manager is responsible for monitoring power consumption, Resource Load and Power Manager monitors the current status of the resources. This method optimises both energy consumption and computing costs, thereby enhancing performance and reducing environmental impact in fog-like scenarios.

This study addresses the problems of work distribution and resource allocation in fog computing by providing a layer fit algorithm and a Modified Harris-Hawks Optimisation (MHHO) strategy. Both of these are discussed in the paper. The proposed solutions cut expenses, make the most efficient use of resources, and distribute the load evenly across cloud and fog levels. The Internet of Things (IoT) enables smart devices to generate large amounts of data and computing labour, which makes it difficult to distribute duties in an effective manner. By preventing oversaturation, degraded service, and resource failures, the layer-fitting algorithm ensures that duties are fairly distributed between layers based on the relative importance of those layers.

In addition, a Modified Harris-Hawks Optimisation (MHHO) meta-heuristic approach is presented for assigning the best available resource within a layer to a task. The goals are to decrease Makespan time, task execution cost, and power consumption while increasing resource utilisation in both layers. The proposed layer fit algorithm and MHHO are compared to traditional optimisation algorithms such as Harris-Hawks Optimisation (HHO), Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO), and the Firefly Algorithm (FA) using the iFogSim simulation toolkit.

According to a study by Edward et.al [7], the load balancing objective in cloud computing is essential owing to the complexity that is produced by enormous amounts of data as well as the potential degradation of the entire system in the event that a defect occurs in a connected virtual machine (VM). The research suggests a unique strategy for resource allocation among virtual machines that is based on transfer learning using Fruitfly. This is done in order to overcome the issues that have been presented. When the virtual machines are first established, they are loaded with a variety of user duties; then, in order to balance the load, the weight is dispersed equitably among all linked VMs. Priority is used to determine the order in which tasks are completed, and resources are allocated in accordance with that order. According to this research paper's findings, the Fruitfly-based transfer learning approach appears to be a potentially useful option for load balancing in cloud computing. It solves the problems that are involved with sharing resources and scheduling tasks, which ultimately leads to improved overall performance and enhanced cloud services.

For the purpose of implementing dynamic resource allocation in cloud computing, Praveenchandar and Tamilarasi [8] suggest a new load-balancing approach that they name PBMM. By putting an emphasis on the dynamic allocation and scheduling of resources, this algorithm overcomes difficulties with load balancing, which in turn promotes stability and profitability. It considers factors such as the size of the undertaking and the value of each customer's bid. The paper [9] employs resource tables and task tables to optimise waiting time and minimise average waiting time and response time for special users. The objective is to maximise profits while improving load-balancing stability, especially by increasing the number of special users. The simulation results demonstrate that the proposed load balancing method accomplishes its goals effectively. It ensures optimum profit by optimising resource allocation and scheduling, and it enhances load balancing stability through increased participation of special users. By taking into account task size, bidding value, and utilising resource and task tables, the proposed algorithm improves stability, reduces waiting periods, and boosts profitability. The findings contribute to the advancement of load balancing techniques in cloud computing environments by emphasising their importance in maintaining service quality and optimising resource utilisation.

A comprehensive answer to the difficulties of limited physical memory, resource underutilization, and ASP profitability in MEC environments. Y. Sun and others have proposed a method [10]. Utilising the Lyapunov optimisation framework and genetic algorithms, the proposed ASP profit-aware solution provides an efficient method for optimising resource allocation, minimising latency, and ensuring long-term profitability. This issue is formulated as a stochastic optimisation problem with ASP profit constraints over the long term. To address this issue, the authors employ the Lyapunov optimisation framework to convert it into a problem involving the optimisation of a specific time slot. Then, they employ genetic algorithms (GA) to create an online heuristic algorithm that approximates near-optimal strategies for each time period. Chetan Kumar et al. [11] propose a method for reducing system latency and increasing ASP profitability. Utilising the edge network's available resources efficiently and minimising latency over time by simultaneously optimising application loading, assignment allocation, and compute resource allocation. It also attracts a greater number of ASVs by enabling them to attain their desired profitability.

Devi et al. [12] proposed a security model based on deep learning to optimise job scheduling while considering security factors into account. The study analyses it and compares it to other conventional scheduling techniques to demonstrate its effectiveness. The research demonstrates the capability of deep learning algorithms to address security issues during task scheduling and emphasises the need for security in cloud computing. This paper contributes to the progress made in cloud computing scheduling by presenting a security approach based on deep learning.

The findings highlight the significance of security concerns in cloud computing environments and provide significant information on the performance of the proposed model compared to existing scheduling techniques.

The paper by Abhikriti and Sunitha [13] examines the significance of scheduling algorithms in cloud computing with regard to the optimisation of resources and the reduction of Makespan time. Despite the fact that existing scheduling algorithms are primarily concerned with reducing Makespan time, they frequently cause load imbalances and wasteful resource utilisation. This research presents a unique Credit-Based Resource Aware Load Balancing Scheduling algorithm (CB-RALB-SA) as a potential solution to the problems described above.

The CB-RALB-SA algorithm's objective is to achieve a balanced distribution of tasks according to the capabilities of the resources available. It assigns weights to jobs using a credit-based scheduling approach, and then maps those tasks to resources after taking into account the load those resources are under and the processing power they possess. The FILL and SPILL functions of the Resource Aware and Load (RAL) approach, which makes use of the Honey Bee Optimisation heuristic algorithm, are used to carry out this mapping.

In [14], the difficulties of load balancing in cloud computing as well as the significance of locating optimal solutions are explored. Swarm Intelligence (SI) is presented as a potential solution for load balancing, and it is contrasted with several different algorithms, including genetic algorithm, ACO, PSO, BAT, and GWO. The algorithms known as Particle Swarm Optimisation (PSO) and Grey Wolf Optimisation (GWO) are the primary foci of this study's investigation.

Federated clouds have emerged as a combination of private and public clouds to facilitate secure access to data from both categories of clouds. However, the procedure of authorization and authentication in federated clouds can be complex.

A new algorithm called the Secured Storage and Retrieval Algorithm (ATDSRA) has been suggested [15] with the intention of providing private and public users of cloud databases with the ability to store and retrieve data in a secure mannerIn addition to the Triple-DES ciphering used for encryption, it combines encrypted data using data merging and aggregation algorithms. In addition, the CRT-based Dynamic Data Auditing Algorithm (CRTDDA) is presented for the purpose of controlling access to federated cloud data and conducting audits of the data.

The ATDSRA algorithm and CRTDDA auditing scheme enhance the security of federated cloud-stored data. The proposed model addresses the complex task of authorization and authentication in cloud computing by providing secure storage, retrieval, and auditing capabilities, thereby enhancing the overall security of federated cloud environments.

In order to solve the problem of load balancing, the authors of [16] propose a combined method namely firefly and BAT. This strategy makes use of the advantages offered by both rapid convergence and global optimisation. This strategy tries to improve both the effectiveness of the system and the distribution of its resources. The findings of the research show that promising results can be achieved in terms of globally optimised quick convergence and decreased total reaction time.

Saydul et al. [17] discuss the significance of task scheduling in cloud computing and how it affects resource utilisation and service performance. It emphasises the need for efficient job scheduling strategies to prevent resource waste and performance decline. Examining various task scheduling techniques within the context of cloud and grid computing, the research identifies their successes, challenges, and limitations. A taxonomy is proposed to classify and analyse these techniques, with the goal of bridging the gaps between existing studies and offering a conceptual framework for more efficient job scheduling in cloud computing.

A priority-based scheduling approach is proposed by Junaid et al. [18] as a method for achieving equitable task scheduling in cloud computing environments. The method's goals are to make the most efficient use of available resources and to boost overall performance. The study throws up questions that could be investigated further in the future and could lead to more effective scheduling tactics. Academicians, policymakers, and practitioners can all benefit from using the priority-based scheduling technique to optimise cloud computing configurations. This is because the technique leads to the creation of more efficient task scheduling strategies.

Ashawa et al. [19] research focuses on resource allocation in large-scale distributed computing systems, specifically in the context of cloud computing. The goal is to maximise overall computing efficiency or throughput by effectively allocating resources. Cloud computing is distinguished from grid computing, and the challenges of allocating virtualized resources in a cloud computing paradigm are acknowledged.

Using the LSTM (Long-Short Term Memory) algorithm, a dynamic resource allocation system is implemented in this study. This system analyses application resource utilisation and determines the optimal allocation of additional resources for each application. The LSTM model is trained and evaluated in simulations that approximate real-time. In addition, the study discusses the incorporation of dynamic routing algorithms designed for cloud data centre traffic. The proposed resource allocation model [20] is compared to other load-balancing techniques and demonstrates improved accuracy rates and decreased error percentages in the average request blockage probability under heavy traffic loads. Compared to other models, the proposed method increases network utilisation and decreases processing time and resource consumption. For load balancing in energy clouds, it is suggested that future research investigate the implementation of various heuristics and machine learning techniques, specifically the use of firefly algorithms.

Cloud computing provides a variety of services to consumers, but the increased demand also increases the probability of errors. Several fault tolerance techniques have been devised in order to mitigate the impact of errors. The paper [21] describes a multilevel fault tolerance system designed to improve cloud environments' reliability and availability. The proposed system has two levels of operation. In the first level, a reliability assessment algorithm is used to determine which virtual devices are trustworthy. By evaluating the dependability of virtual machines, the system guarantees that tasks are assigned to the most dependable resources, thereby minimising the possibility of errors. By replicating data and distributing it among a number of physical or virtual devices, the replication method ensures that the data will always be accessible at the second level. This technique ensures that even if one copy becomes inaccessible or malfunctions, other replicas can still be accessed, hence protecting the availability of data even if that one copy becomes unavailable or malfunctions. This multilevel fault tolerance method improves fault tolerance in real-time cloud environments by combining reliable virtual machine identification with data replication. The result is a reduction in mistakes and an increase in the efficiency of cloud services.

Inattention to federated cloud environments Federated clouds, which combine private and public clouds, present unique challenges for authorization, authentication, and secure data storage. The absence of efficient algorithms and models to resolve these complexities and improve the overall security of federated cloud environments represents a research gap.

Table 1. Comparison of Existing Methods.

Citation	Technique	Advantage	Limitations
Ali Asghar Heidari et al., 2019	Meta-heuristic algorithms (e.g., HHO)	Effective optimization capabilities, near-optimal solutions	HHO implementation in cloud computing is largely unexplored
K Vinoth Kumar & A Rajesh, 2022	Multi-objective optimization strategy	Load balancing method for data centers, efficient algorithm, adjustable to different workload circumstances	Limited discussion on the algorithm's resilience and adaptability
Shafiq D A et al., 2021	CRLBT strategy with TM-PSO and TM-Firefly	Improved response time, task rejection ratio, CPU utilization rate, and network throughput	--
Oduwole O A et al., 2022	DEER strategy	Increased efficacy, reduced energy consumption, decreased environmental impact	Limited discussion on the dynamic nature of fog environments
Rehman A U et al., 2020	Layer fit algorithm and MHHO strategy	Enhanced performance metrics, reduced costs, maximized resource utilization, load balancing in fog computing	Comparison with traditional optimization algorithms
Edward G, Geetha B & Ramaraj E, 2023	Fruitfly-based transfer learning	Improved load balancing, resource sharing, and task scheduling	Limited discussion on the algorithm's performance compared to others

doi:10.20944/preprints202307.1612.v1

Praveenchandar & Tamilarasi, 2022	PBMM algorithm	Dynamic resource allocation, improved load balancing stability and profitability	--
Narwal A & Sunita D, 2023	CB-RALB-SA algorithm	Balanced distribution of tasks, efficient load balancing, honey bee optimization	--
Al Reshan M.S. et al., 2023	SI, PSO, and GWO algorithms	Potential load balancing solutions, comparison with other algorithms	--
Sermakanu A.M., 2020	ATDSRA and CRTDDA algorithms	Secure storage and retrieval, restricted data access	Limited discussion on the algorithm's performance compared to others
Saydul A.M. et al., 2022	Task scheduling techniques analysis	Efficiency of job scheduling, resource utilization, performance improvements	Taxonomy proposed for classification and analysis of scheduling techniques
Ashawa M et al., 2022	LSTM-based dynamic resource allocation	Maximization of computing efficiency, improved resource allocation	Limited discussion on the comparison with other allocation techniques
Hariharan B, 2020	Resource allocation model	Increased accuracy, decreased error percentages, improved network utilization	Suggested further research on heuristics and machine learning techniques

2.1. Summary of the Literature review

The purpose of this literature review is to investigate the difficulties associated with load balancing and resource allocation in cloud computing settings and to investigate the various techniques that researchers have offered to address these concerns. In cloud computing, it places a primary emphasis on the utilisation of meta-heuristic methods, such as the Harris hawk optimisation (HHO) algorithm, for the purposes of load balancing and resource allocation.

Well-known meta-heuristic algorithms that are employed in cloud computing, such as genetic algorithms, particle swarm optimisation, ant colony optimisation, and simulated annealing, are investigated in one study. It examines their advantages and disadvantages, as well as the uses they have in load balancing and resource distribution.

A method for load balancing in data centres that makes use of a multi-objective optimisation strategy has been proposed in one study. In yet another piece of research, the issue of load distribution in big-data cloud systems is investigated via the lens of the Central-Regional Architecture Based Load Balancing Technique (CRLBT). In order to manage the flow of information over complicated networks for the Internet of Things (IoT), a technique known as DEER has been presented as a solution for load balancing in fog computing settings. This comes as the importance of fog computing in this area continues to grow.

In addition to that, a paper provides a layer fit technique as well as a Modified Harris-Hawks Optimisation (MHHO) strategy to solve the job distribution and resource allocation issues that are inherent in fog computing. In this article, a novel load balancing algorithm known as PBMM and a load balancing technique based on transfer learning utilising Fruitfly are discussed for the purpose of dynamic resource allocation in cloud computing.

The introduction of a deep learning-based security model and a Credit-Based Resource Aware Load Balancing Scheduling algorithm brings to light the necessity of security in cloud computing scheduling. Both of these innovations are designed to balance and distribute workloads. In addition to this, the paper delves into the topic of swarm intelligence algorithms and suggests a Secured Storage and Retrieval Algorithm (ATDSRA) for usage in federated cloud environments.

In addition, research on task scheduling, priority-based scheduling strategies, resource allocation in distributed computing systems, dynamic resource allocation utilising the LSTM algorithm, fault tolerance systems, and reliability assessment in cloud environments are all covered in this review.

This literature review sheds light on the existing body of knowledge about load balancing and resource allocation in cloud computing. Additionally, it draws attention to the potential of meta-heuristic algorithms and other methodologies to effectively solve these difficulties.

2.2. Research Gap

The research gap in the literature necessitates further exploration and development of load balancing and resource allocation techniques in cloud computing, taking into consideration specific environments, security concerns, combined optimisation techniques, and dynamic resource allocation strategies. In addition, there is a need for a comprehensive analysis, classification, and evaluation of existing techniques in order to provide insight and direction for the development of more effective solutions.

- Limited investigation of the use of the Harris hawk optimisation (HHO) algorithm in cloud computing load balancing and resource allocation:

  Although meta-heuristic algorithms, such as HHO, have demonstrated promise in solving optimisation problems, their application in cloud computing load balancing and resource allocation is primarily unexplored.

- Lack of comprehensive analysis and comparison of meta-heuristic algorithms:

  Load balancing techniques tailored to specific cloud computing environments, such as fog computing and big-data cloud systems, are required. Existing research frequently disregards the distinctive characteristics and difficulties of these environments.

- Insufficient focus on load balancing strategies for specific cloud computing environments:

  Although load balancing and resource allocation are essential for optimising system performance, guaranteeing security in the scheduling process is equally crucial. In the context of task scheduling, the incorporation of security factors, such as deep learning-based security models, requires further investigation.

- Inadequate attention to security concerns in load balancing and resource allocation:

  Equally as essential as load balancing and resource allocation for optimising system performance is the security of the scheduling process. In the context of task scheduling, the incorporation of security factors, such as deep learning-based security models, requires further investigation.

- Lack of emphasis on federated cloud environments:

  Federated clouds, which combine private and public clouds, present unique authorization, authentication, and secure data storage challenges. The absence of efficient algorithms and models to resolve these complexities and improve the overall security of federated cloud environments represents a research gap.

- Limited exploration of combined optimization techniques:

  There is a need for research that investigates the effectiveness of combining various optimisation techniques, such as the firefly and BAT algorithms, to improve load balancing and resource allocation in cloud computing.

- Incomplete understanding of the impact of scheduling on resource utilization and system performance:

  Task scheduling has a substantial impact on resource utilisation and overall system performance. Further study is required to develop more effective scheduling techniques that maximise resource utilisation and minimise Makespan time while taking load balancing and performance tradeoffs into account.

- Inadequate investigation of dynamic resource allocation:

  Dynamic resource allocation necessitates techniques that optimise computing efficiency, particularly in large-scale distributed computing systems. Dynamic resource allocation models, such as LSTM-based algorithms, and their integration with dynamic routing algorithms for cloud data centre traffic require additional study.

3. **Objectives and Problem Statement**

In the world of cloud computing, the issue of overloading manifests itself when jobs arrive in an unpredictable order. It is common for some resources to become significantly loaded as a result of the arbitrary utilisation of the CPU, while other resources continue to remain idle. In order to solve this problem, load balancing, which involves the allocation of work over a network according to the number of virtual machines (VMs) or central processing units (CPUs), has been put into use. The primary objective of cloud computing is to guarantee optimal resource utilisation, which will ultimately result in enhanced system performance and a reduction in response time.

In addition, load balancing algorithms face obstacles such as the high cost of hardware and bottlenecks in the system. However, these approaches have overlooked the consideration of a resource's ability to accomplish tasks and the fulfilment of user requirements. As a result, resource utilisation has been further reduced, and cloud systems have been presented with challenges. In addition, even if these methods have improved superiority, they have not given sufficient scalability or response speed. Also, the substantial costs associated with hardware and system bottlenecks have surfaced as key concerns in the approaches of load balancing. An effective Multi-Objective Meta-Heuristic Dynamic load balancing and optimisation method has been implemented in the cloud environment to handle load balancing concerns. This was done to help overcome the challenges that have been presented.

4. Model Development

In cloud computing, the process of load balancing and scheduling is handled by a N number of virtual machines (VMs).

$$VM = \sum_{i=1}^{N} Vm_i \tag{1}$$

In order to ensure that the workload is distributed fairly and equitably across all of the servers, each VM has been given a specific server, hence there are M servers as well and

$$S = \sum_{s=1}^{M} S_s \tag{2}$$

each Virtual Machine has its own independent collection of resources. Memory, central processing unit, disc space, and bandwidth are all included in the resources.

$$\begin{aligned} R_{vm_1} &= \{R_{1,1}, R_{1,2}, \ldots R_{1,j}\} \\ R_{vm_2} &= \{R_{2,1}, R2, \ldots R_{2,j}\} \\ R_{vm_n} &= \{R_{n,1}, R_{n,2}, \ldots R_{n,j}\} \end{aligned} \tag{3}$$

Let us assume that there are many numbers of jobs in the queue which needs the resources.

$$J = \{J_1, J_2, ..., J_n\} \qquad (4)$$

These jobs are loaded into the cloud server by many cloud users, say U. Where

$$U = \{U_1, U_2, ..., U_n\} \qquad (5)$$

The virtual machines (VMs) are assigned jobs with the intention of achieving an optimal load distribution. The architecture is made up of limited number of servers, one of which is designated to be the central server and is in charge of receiving requests from virtual machines (VMs). Each server is able to process requests, which brings the VMs up to par with the most powerful server. The goal is to design a better scheduler that can assign jobs to virtual machines (VMs) in a way that ensures load balancing across all of them. To accomplish the objective of optimising resource utilisation, user characteristics such as request size, number of CPU requests, and number of requests sent per time interval will be employed. To combine virtual machines (VMs) for cloud load balancing, you'll need hosts with a certain fitness value and their own resources. The primary objective is to develop a superior scheduler capable of distributing work to virtual machines (VMs) in accordance with available resources.

In the scope of this study, load balancing is optimised in terms of reaction time, cost, and resource utilisation. The objectives are to decrease reaction time, cut costs, increase resource utilisation, and increase overall utilisation. Load balancing consists of numerous components. The objective of extensive research and concentrated efforts to reduce reaction times, expenses, and resource consumption is to enhance efficiency and effectiveness. The output can be increased through more efficient distribution and allocation of resources. Utilising all of a system's available resources to their maximum capacity helps eliminate bottlenecks and inefficiencies. To determine the amount of resources utilised, use equation (6).

$$\begin{aligned} R_{utilization} &= \left(\frac{R_{used}}{R_{total}}\right) \times 100 \\ R_{overall-utilization} &= \sum_{R=1}^{R_{resources}} \left(\frac{OU_{R_{resources}}}{R_{total}}\right) \\ Where,\ OU_{R_{resources}} &= Overall\ Utilization\ of\ the\ resources \end{aligned} \qquad (5)$$

By enhancing resource utilisation, we can maximise the system's capabilities and eliminate any potential bottlenecks or inefficiencies. To calculate resource usage, use the accompanying equation (6).

$$\begin{aligned} T_{exe} &= T^{J}_{Completion} - T^{J}_{Arrival} + T^{Transmission}_{delay} \\ J^{T}_{exe} &= J^{T}_{start} - J^{T}_{arrival} \\ T_{response} &= T_{exe} + J^{T}_{exe} \end{aligned} \qquad (6)$$

When evaluating these various objectives, we discover that cost, resource utilisation, and response time are interrelated. To attain the goal of a well-balanced and highly effective load balancing system, extensive optimisation and fine-tuning are required. The following algorithm 1 computes the cost of a single work by considering the time required to initiate the task, the time required to conclude the project, and the required resources.

It determines the execution time by first subtracting the start time from the end time and then dividing the difference by two. The cost is then determined by multiplying the duration of execution by the quantity of resources utilised throughout the activity.

Algorithm 1: Calculate the Cost

```
$Function\ \mathrm{Cos}\,t()$
$Init : \mathrm{Cos}\,t = 0, \mathrm{C}_{Total} = 0, T_{current} = 0$
$Function\ \mathrm{Cos}\,tCalculation()$
  $T_{exe} = T^{J}_{Completion} - T^{J}_{start}$
  $\mathrm{Cos}\,t = T_{exe} \times R^{J}_{usage}$
$return\ \ \mathrm{Cos}\,t$
$Function\ JobScheduling()$
  $S^{J} = \bigcup_{i=1}^{n} J^{T}_{arrival}$
      $repeat\ \ for\ all\ \ jobs\ in\ the\ queue$
          $if\ \left(T^{J}_{arrival} > T^{C}\right) then$
            $T^{C} = \ T^{J}_{arrival}$
          $J^{T}_{start} = T^{C}$
          $J^{T}_{Completion} = Exe(J)\ //Execute\ the\ job\ and\ get\ the\ time$
          $J_{\cos t} = \int_{j=1}^{J} R^{j} + \sum \left(J^{T}_{start}, J^{T}_{Completion}\right)$
          $\mathrm{C}_{Total} = Total\,\mathrm{Cos}\,t + J_{\cos t}$
          $T_{current} = T^{J}_{Completion}$
$\mathrm{Re}\,turn\ \mathrm{C}_{Total}$
```

5. Meta-Heuristic Harris Hawk Algorithm

The Harris Hawk Optimisation Algorithm (HHO) is a problem-solving optimisation algorithm that was inspired by nature and is used in cloud computing to address resource allocation issues. It was modelled after the Harris hawk's strategy of working together while hunting, which boosts their overall efficiency. The HHO algorithm's primary objective is to improve overall system performance by optimising resource utilisation across a variety of roles and responsibilities. It requires multiple steps to effectively distribute resources while simultaneously cutting expenses and improving efficiency.

5.1. Initialization

The initialization phase of the Harris Hawk Optimisation (HHO) algorithm is the first stage in the optimisation process. During this stage, viable solutions are set up, and initialization parameters for subsequent iterations are determined.
The size of the population, designated by the letter P_N, and a random method of resource distribution is devised for each person in the population, from one to N, in order to distribute the resources. In order to accomplish this, random values need to be assigned to the allocation parameters, which can include things like the number of virtual machines (VMs), CPU cores, memory, disc space, and so on. Each solution illustrates a feasible method for allocating resources to perform various cloud computing activities and is shown in Algorithm 2.

Algorithm 2 : Population Initialization

```
$Function\ PopulationInitialization()$
$Init : P_N, B_L, B_U$
$P_N = 0$
$for\ i = 1\ to\ N$
    $//Create\ new\ Individual$
    $I_i = New(I)$
     $for\ each\ j$
        $//$Generate a random number that falls somewhere between the upper and lower bounds.
        $I_j = Rand(B_L, B_U)$
        $//$Include the individual on the list of individuals that collectively make up the population.
        $P_N = P_N + I_j$
$Return\ P_N$
```

5.2. Fitness Evaluation

Evaluate how well each solution in the population fits to the requirements. When discussing cloud computing, fitness is typically discussed in relation to measures such as response time, throughput, energy consumption, or cost. The fitness function provides a quantitative assessment of the quality of a resource allocation strategy that is founded on these metrics. Algorithm 3 is used to calculate the fitness value.

Algorithm 3: Fitness Evaluation

```
$Function\ FitnessEvaluation()$
  $for\ each\ I_i \in P_N\ \ //for\ each\ individual\ in\ population$
  $//$Manage the distribution of resources and execute the tasks that are assigned.
  $I_i = \sum_{i=1}^{n} R_i$ //This allocates the available resources (such as virtual machines,
           //CPU processors, memory, and disc space) based on the allocation
           //parameters of the individual.
  $Exec\left(I_i^{Task}\right)$
  $F = \iiint T^c \bullet R_{utilization} \bullet R_{efficiency}\ \ //F - fitness$
$F_V\left(I_i\right) = F\ \ //\ F_V = Fitness\ Value$
```

5.3. Leader selection

During the leader selection phase of the Harris Hawk Optimisation (HHO) algorithm, the goal is to find the solutions within the population that have demonstrated the highest levels of performance, which are referred to as leaders. The leaders guide the search process and influence the exploration and exploitation of the solution space. The algorithm to select the leader is as shown in algorithm 4.

Algorithm 4: Leader Selection

```
Function LeaderSelection()
  Input: $P_N, F_v$        // Population, Fitness
  $P_{Size} = Len(P_N)$   // Population Length
  $P_N = S_{Desc}(P_N)$    // Sort the population in Desecding order based on their fitness value
  $L = [P_N[1]]$           // Select the best (top) population to serve as the initial leader
  for $i = 2$ to $i \le P_{Size}$
    if $(F_v[i]) > F_v[L]$
      $L = F_v[i]$
Return L
```

The fitness value of each solution in the population is determined by the function as it iteratively processes its way through all of the solutions in the population. The result that achieves the greatest overall fitness is deemed to be the leader.

5.4. Exploration

In the Harris Hawk Algorithm's (HHO) Exploration phase, a simulation of the hunting behaviour of Harris's Hawks is used to explore the search space and uncover potential solutions to an optimisation problem. During this stage, a population of hawks will be created, and both their placements and their fitness values will be assessed.

Harris hawk's cooperative hunting behaviour as a model for exploring the space available for solutions is utilised in this phase. As part of this process, the positions of the solutions in the population will be updated in an effort to discover more effective solutions. In order to direct the process of exploration, local search, a global search, or a combination of the two may utilised and this process in explained in Algorithm 5.

The first step of the Exploration phase is to generate the initial positions of the hawks inside the search space using a randomization algorithm. The fitness of each hawk is determined based on its location, which is a representation of how well it performs in terms of finding the optimal solution to the problem. The algorithm keeps track of a global best position and fitness, which are both initialised with the values of the hawk in the initial population that has the highest level of fitness. When a hawk makes a discovery throughout the exploration process that leads to a better location, this global best position is immediately updated.

A hunting strategy is implemented for each individual hawk in the population so that they can investigate and potentially improve their position in each iteration of the model. In the course of the hunting strategy, four hawks at random are chosen from the population and given the designations H_1, H_2, H_3, and H_4. The hawk that is currently being considered is not one of these four. To determine how far apart hawks H_1 and H_2 are from one another, a distance metric, such as the Euclidean distance, is applied to the calculation. The direction vector is determined by determining the difference in position between H_1 and the current hawk. This difference is the direction in which the hawk should move, and the location of the hawk is then updated to reflect this new information. After the Exploration phase of the algorithm has been completed, it will then return the global best position and fitness as the final result. These two values will reflect a probable optimal solution to the optimisation issue of the resource allocation in cloud computing.

doi:10.20944/preprints202307.1612.v1

Algorithm 5: Exploration

```
Function Exploration()
Initialize:
   $P_N^H$   // No. of Hawks in the population
   $I, I_{Max}$   // Maximum Iteration
   $_{\min}P_f^u, _{\max}P_f^u$   // Minimum and Maximum update factor values for each individuals
   $_{pos}H[]$  // Haw's position array
   $F_v[]$  // Fitness Value Array
   $U_f$  // Update Factor
   $D_{direction}$  // Direction of the Hawk
   $B_L, B_U$  // Lower and Upper Boundaries
   Call PopulationInitialization() // Algorithm 2
   $I_j = Rand(U_L, U_B)$  // Randomly Generate the position
   Call FitnessEvaluation() // Algorithm 3
   $P_{optimal}^H = \lim_{\delta x \to N}\left[{}_{pos}H\left[P_N^H\left[F_v[]\right]\right]\right]$  //Optimal Position of the Hawk in the initial population
   $F_{optimal}^H = \lim_{\delta x \to N}\left[F_v\left[P_N^H\left[F_v[]\right]\right]\right]$  ///Optimal Fitness Value of the Hawk in the initial population
  $I = 1$
  Repeat Until $(I \le I_{Max})$
     $for\ i = 1\ to\ P_N^H$
     // Adjust the Hariss Hawk movement
     Choose four Hawks $(H_1, H_2, H_3, H_4)$ at random from $P_N^H \notin P_N^H[i]$
     $D_{euclidean} = Euclidean\_Distance\left(P_N^H[H_1] - P_N^H[H_2]\right)$ // Distance between the Hawks
     $if\ (F_v[H_1] > F_v[H_2])$
        $U_f = {}_{\max}P_f^u \times e^{\left(\frac{-I}{I_{Max}}\right)}$
     else
        $U_f = {}_{\min}P_f^u \times e^{\left(\frac{-I}{I_{Max}}\right)}$
   $D_{direction} = P_{optimal}^H[H_1] - P_N^H[i]$
   $P_N^H[i] = P_N^H[i] + D_{direction}$
  //Set the position's boundaries inside the boundaries of the search space
  $P_N^H[i] = \lim\left(P_N^H[i], B_L, B_U\right)$
  $F_v[i] = F_v\left[P_N^H[i]\right]$  ////Evaluate the fitness of the new position
  $if\ \left(F_v[i] > F_{optimal}^H\right)$
   $F_{optimal}^H = F_v[i]$
   $P_{optimal}^H = P_N^H[i]$
  $I = I + 1$
Return$\left(P_{optimal}^H, F_{optimal}^H\right)$
```

6. Results and Discussion

6.1 Simulation and Results

The results of the simulation were an extremely helpful resource when doing an analysis to evaluate the efficiency of the proposed method. In order to make this analysis easier, the powerful CloudSim simulation software was utilized. CloudSim 4.0 excels in the provision of modeling and simulation activities, in addition to providing cloud computing services and the ability to test applications. The simulation was carried out on a computer that featured an Intel Core i7 CPU, 12 gigabytes of Random Access Memory (RAM) and the central processing unit (CPU) ran at a frequency of 2.80 gigahertz speed used Windows 10.

The proposed work has been compared with four other similar studies, including HHO [22], MRFO [23], QMPSO [24], and MMHHO [25], and that the effectiveness of the MDLB-HHO technique has been evaluated. According to the findings, the suggested MDLB-HHO performed similarly to or even better than the other works in a number of different metrics, additionally, the proposed MDLB-HHO maintained a comparable level of performance in some of the other metrics.

The performance comparison of proposed and existing approaches in terms of energy utilization with varying numbers of tasks and virtual machines is shown in Figure 1 and Figure 2.

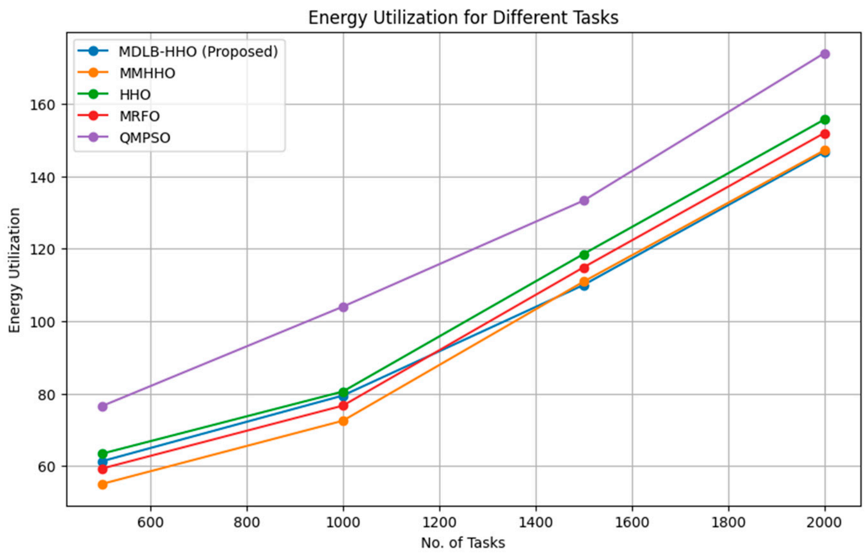

Figure 1. Energy Utilization of different algorithms with variable number of tasks.

In the same manner, the Makespan is an essential metric in cloud computing that calculates the overall amount of time necessary to finish a collection of activities or tasks in a cloud-based system. It is the amount of time that elapses between the beginning of the first task in a certain workload and the finish of the final task in that workload. When it comes to cloud computing, the major goal is to reduce the Makespan as much as possible because this factor directly affects the system's efficiency and performance. It is crucial to cut down on the Makespan since this results in speedier completion of activities, increased resource usage, and higher overall system productivity. Cloud service providers are able to optimize resource allocation, improve customer satisfaction, and achieve cost-effectiveness when they reduce the Makespan as much as possible. Researchers in the field of cloud computing are constantly looking into novel scheduling algorithms, resource management approaches, and optimization tactics in order to maximize the maketime of cloud applications. Cloud computing platforms that have achieved a minimal Makespan are able to provide services that are both quicker and more dependable, hence enhancing the overall user experience and making the most efficient use of available resources. The Makespan of the proposed system for different number of tasks and VMs are shown in Figure 3 and Figure 4.

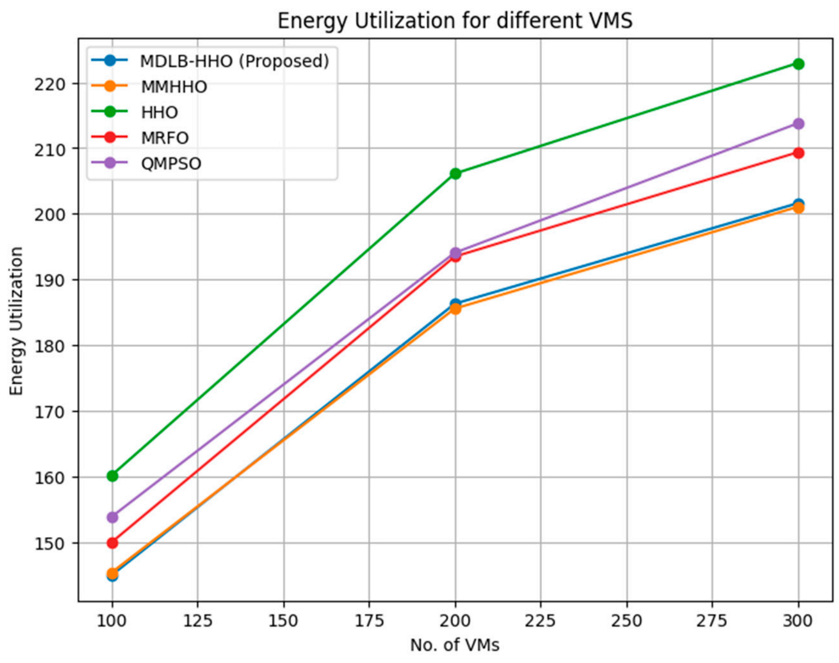

Figure 2. Energy Utilization of different algorithms with variable number of VMs.

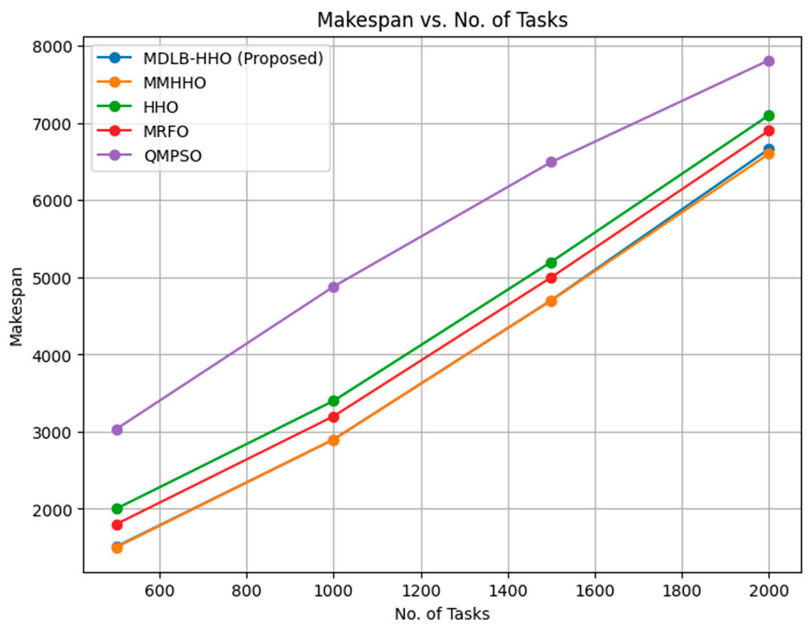

Figure 3. Makespan of different algorithms with variable number of tasks.

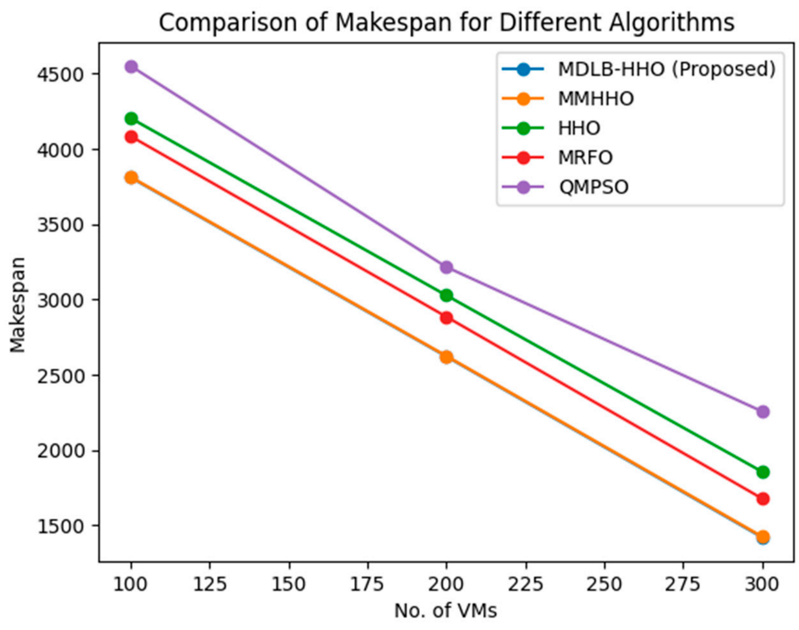

Figure 4. Makespan of different algorithms with variable number of VMs**.**

6.2. Discussion

Figure 1 shows the energy used by different algorithms for different tasks. Algorithms considered include MDLB-HHO (Proposed), MMHHO, HHO, MRFO, and QMPSO. After 500, 1000, 1500, and 2000 tasks, each method is evaluated for energy use. The results show intriguing commonalities in energy utilization across algorithms and job counts.

The MDLB-HHO (Proposed) method consumes more energy as workloads increase. 61.29 for 500 jobs rises to 79.43 for 1000 tasks, 109.99 for 1500 tasks, and 146.72 for 2000 duties. This shows that effort increases energy consumption proportionally. The MMHHO algorithm uses more energy as tasks increase. This is an algorithm characteristic. After 500 increments, it hits 72.47, 110.91, and 147.18 for 1000, 1500, and 2000 tasks. HHO uses slightly more energy than MDLB-HHO (Proposed) and MMHHO for all task numbers. For 500, 1000, 1500, and 2000 tasks, the numbers are 63.38, 80.54, 118.55, and 155.69.

MRFO energy consumption matches MDLB-HHO (Proposed), MMHHO, and HHO. 500, 1000, 1500, and 2000 tasks provide 59.27, 76.64, 114.86, and 151.97. QMPSO uses more energy across all job counts than the other algorithms tested. For 500, 1000, 1500, and 2000 tasks, it consumes 76.54, 104, 133.3, and 174.03 more energy than the other methods. Lowest to highest. These data show how algorithms consume energy at different work loads. MDLB-HHO (Proposed), MMHHO, HHO, and MRFO all use energy similarly. QMPSO uses more energy on average than the other algorithms. Understanding these energy usage variations can help choose an algorithm for a certain work based on performance and energy efficiency.

Figure 2 shows the energy usage of algorithms over varying numbers of virtual machines (VMs). MDLB-HHO (Proposed), MMHHO, HHO, MRFO, and QMPSO algorithms are being evaluated. As VMs expand, energy usage increases. As VMs increase, energy usage rises across all algorithms. More virtual machines require more processing resources, which explains this tendency. The algorithms MDLB-HHO (Proposed) and MMHHO use energy similarly. Both techniques use more energy from 100 to 300 VMs. MMHHO uses slightly more energy than MDLB-HHO (Proposed) for all VM amounts. MDLB-HHO (Proposed) and MMHHO use less energy than HHO for all VM amounts. HHO may take more computational resources and energy to execute tasks due to this discrepancy. MRFO and QMPSO algorithms use less energy than HHO. As VMs rise, both algorithms need more energy. MRFO uses slightly less energy than QMPSO. These findings emphasize the need for energy-efficient cloud virtual machine management methods. Energy efficiency minimizes operational expenses and carbon footprint.

Figure 3 shows the Makespan for several methods at varying task counts. MDLB-HHO (Proposed), MMHHO, HHO, MRFO, and QMPSO algorithms are under consideration. 500-task increments increase to 2000 tasks. The MDLB-HHO (Proposed) and MMHHO algorithms have the lowest Makespan values across all task numbers. Both algorithms do tasks efficiently and effectively. Both techniques increase Makespan steadily as the number of tasks grows. HHO, however, produces slightly higher Makespan values than MDLB-HHO (Proposed) and MMHHO. Makespan also increases with job count. HHO performs well and completes jobs, however not as efficiently as the other algorithms. MRFO performs comparably to MDLB-HHO (Proposed) and MMHHO. It manages a variety of jobs more efficiently than HHO, with a lower makespan. Finally, the QMPSO algorithm has the longest makespan. As tasks increase, makespan increases, suggesting scalability issues. QMPSO still finishes tasks, although it takes longer. The MDLB-HHO (Proposed) and MMHHO algorithms consistently have the lowest makespan values across all task numbers. These algorithms excel at managing tasks in a time-sensitive context. The HHO and MRFO algorithms perform well, despite their slightly greater makespan values. Finally, the QMPSO algorithm yields reasonable results but has a longer makespan than the others, suggesting room for optimization. These results illuminate each algorithm's strengths and weaknesses, helping researchers and practitioners choose a task management algorithm that balances completion time and efficiency.

Figure 4 shows makespan values for different VM counts. Data analysis yields many findings. At each VM count, MDLB-HHO (Proposed) and MMHHO have similar makespan values. This shows both strategies minimize makespan similarly. HHO has greater makespan values than MDLB-HHO (Proposed) and MMHHO, but it outperforms MRFO and QMPSO across all VM counts. HHO is effective, but not as well as MDLB-HHO (Proposed) and MMHHO. MRFO and QMPSO had the largest makespan values of the evaluated algorithms, regardless of VM count. Based on this dataset, MRFO and QMPSO may not be the best solutions for lowering makespan in parallel computing systems. All algorithms decrease makespan as VMs increase. Task distribution across more VMs improves load balancing and efficiency.MDLB-HHO (Proposed) and MMHHO algorithms minimize makespan better, according to data.

7. Conclusion

This paper introduced a multi-objective meta-heuristic dynamic load balancing and resource allocation technique in cloud computing utilizing the Harris hawk optimization (MDLB-HHO) algorithm. The proposed load balancing method reduces Makespan time and improves resource utilization. The load balancing challenge was formulated as a multi-objective optimization problem and solved using the MDLB-HHO algorithm to dynamically assign jobs to VMs based on workload distribution and resource utilization. The MDLB-HHO technique allowed the hawks to collaboratively search the solution space and discover the best task allocations through iterative interactions and position updates. The load balancing solution for dynamic cloud environments addresses numerous objectives and hawks' cooperative hunting behavior. The suggested cloud computing multi-objective optimization strategy incorporates cost, resource utilization, and Makespan time. Optimizing these criteria improves cloud computing system operation and resource utilization. This research concluded with an HHO algorithm-based multi-objective dynamic load balancing and resource allocation approach for cloud computing. The proposed method outperformed other methods in the experiments. Future research may include applying the MDLB-HHO algorithm in specific cloud computing environments, addressing security concerns in load balancing and resource allocation, and integrating multiple optimization techniques to improve system performance.

References

1. Ali Asghar Heidari; Seyedali Mirjalili; Hossam Faris; Ibrahim Aljarah; Majdi Mafarja and Huiling Chen. Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems* **2019**, *97*, pp.849-872.
2. Author 1, A.B.; Author 2, C.D. Title of the article. *Abbreviated Journal Name* **Year**, *Volume*, page range
3. Kethineni Vinod Kumar and Rajesh, A. Multi-objective load balancing in cloud computing: A meta-heuristic approach. *Cybernetics and Systems* **2022**. DOI: 10.1080/01969722.2022.2145656.
4. Shafiq, D.A.; Jhanjhi, N.Z.; Abdullah, A. and Alzain, M.A. A load balancing algorithm for the data dentres to optimize cloud computing applications. *IEEE Access* **2021**, *9*, pp.41731-41744.
5. Oduwole, O.A.; Akinboro, S.A.; Lala, O.G. and Olabiyisi, S.O. An enhanced load balancing technique for big-data cloud computing environments. *Transactions of the Royal Society of South Africa* **2022,** *77*(3), pp.219-236.
6. Rehman, A.U.; et al. Dynamic energy efficient resource allocation strategy for load balancing in fog environment. *IEEE Access* **2020,** *8*, pp.199829-199839.
7. Yakubu, I.Z. and Murali, M. An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment. *Journal of Ambient Intelligence and Humanozed Computing* **2023,** *14*, pp.2981–2992.
8. Edward Gerald, B.; Geetha, P. and Ramaraj, E. A fruitfly-based optimal resource sharing and load balancing for the better cloud services. *Soft Computing* **2023**, *27*, pp.6507–6520.
9. Praveenchandar, J. and Tamilarasi, A. An enhanced load balancing approach for dynamic resource allocation in cloud environments. *Wireless Personal Communication* **2022**, *122*, pp.3757–3776.
10. Reshmy, A.K. Data mining of unstructured big data in cloud computing. *International Journal of Business Intelligence and Data Mining* **2017**, *13*(1-3), pp.147-162.
11. Sun, Y.; Xie, X.; Wu, F.; Zhang, S.; Xu, S. and Wu, Y. Application loading and computing allocation for collaborative edge computing. *IEEE Access* **2021**, *9*, pp.158481-158495.
12. Chetan Kumar; Sean Marston; Ravi Sen and Amar Narisetty. Greening the cloud: A load balancing mechanism to optimize cloud computing networks. *Journal of Management Information Systems* **2022**, *39*(2), pp.513-541.
13. Devi, K. and Muthusenthil, B. Deep learning-based security model for cloud based task scheduling. *KSII Transactions of Internet and Information Systems* **2020**, *14*(9), pp.3663-3679.
14. Abhikriti Narwal and Sunita Dhingra. A novel approach for credit-based resource aware load balancing algorithm (CB-RALB-SA) for scheduling jobs in cloud computing. *Data and Knowledge Engineering* **2023**, *145*, 102138.
15. Al Reshan, M.S.; et al. A fast converging and globally optimized approach for load balancing in cloud computing. *IEEE Access* **2023**, *11*, pp.11390-11404.
16. Sermakani, A.M. Effective data storage and dynamic data auditing scheme for providing distributed services in federated cloud. *Journal of Circuits, Systems and Computers* **2020,** *29*(16), 2050219.
17. Hariharan, B. A hybrid framework for job scheduling on cloud using firefly and BAT algorithm. *International Journal of Business Intelligence and Data Mining* **2019**, *15*(4), pp.388-407.
18. Saydul Akbar Murad; Abu Jafar Md Muzahid; Zafril Rizal M Azmi; Md Imdadul Hoque; Md Kowsher. A review on job scheduling technique in cloud computing and priority rule based intelligent framework. *Journal of King Saud University - Computer and Information Sciences* **2022**, *34*(6), pp.2309-2331.
19. Junaid, M.; Sohail, A.; Ahmed, A.; Baz, A.; Khan, I.A. and Alhakami, H. A hybrid model for load balancing in cloud using file type formatting. *IEEE Access* **2020**, *8*, pp.118135-118155.
20. Ashawa, M.; Douglas, O.; Osamor, J. et al. Improving cloud efficiency through optimized resource allocation technique for load balancing using LSTM machine learning algorithm. *Journal of Cloud Computing : Advances, Systems and Applications* **2022**, *11*(87).
21. Hariharan, B. WBAT Job scheduler: A multi-objective approach for job scheduling problem on cloud computing. *Journal of Circuits, Systems and Computers* **2020**, *29*(6).
22. Devi, K. Multi level fault tolerance in cloud environment. *Proceedings of the International Conference on Intelligent Computing and Control Systems* **(2017).**, June 15-16; Madurai, India, pp. 824-828.
23. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M. and Chen, H. Harris hawks optimization Algorithm and applications. *Future Generation Computer Systems* **2019**, *97*, pp.849–872.
24. Zhao, W.; Zhang, Z.; Wang, L. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Engineering Applications of Artificial Intelligence* **2020**, *87*, 103300.
25. Jena, U.K.; Das, P.K. and Kabat, M.R. Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences* **2020**, *34*(6,Part A), pp.2332-2342.

26. Mohammad Haris and Swaleha Zubair**.** Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing. *Journal of King Saud University-Computer and Information Sciences* **2022**, *34*(10, Part B), pp.9696-9709.