National College of Ireland

Enhancing Load Balancing in Cloud Computing and Reducing Makespan by using Hybrid Particle Swarm Optimisation Algorithm to Improve Task Scheduling.

MSc Research Project
Cloud Computing

Prathamesh Dattatray Prabhutendolkar
Student ID: x21127352

School of Computing
National College of Ireland

Supervisor: Rashid Mijumbi

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Prathamesh Dattatray Prabhutendolkar
Student ID:	x21127352
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Rashid Mijumbi
Submission Due Date:	15/12/2022
Project Title:	Enhancing Load Balancing in Cloud Computing and Reducing Makespan by using Hybrid Particle Swarm Optimisation Algorithm to Improve Task Scheduling.
Word Count:	7363
Page Count:	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	14th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Load Balancing in Cloud Computing and Reducing Makespan by using Hybrid Particle Swarm Optimisation Algorithm to Improve Task Scheduling.

Prathamesh Dattatray Prabhutendolkar
x21127352

Abstract

Utilizing virtual machines as the resource unit, cloud computing provides customers with a range of computational services through the Internet, including backups, software, databases, and servers. Tasks are distributed between virtual machines in a cloud computing ecosystem, each of which has a varied length, beginning, and processing time. Therefore, distributing these loads equally throughout the available cluster of virtual machines is a critical part. To achieve maximum utilization of the cluster's capabilities and enhance system efficiency, task scheduling strategies must be implemented in a way that helps balance the workload among all VMs. Recently, researchers have introduced nature-inspired algorithms into the field of task scheduling to overcome challenges connected with complexity and to deliver optimum solutions. In this research paper, we present a unique load-balancing method called the Hybrid Particle Swarm Optimization Algorithm to distribute workloads among the available cloud resources in such a way that reduces execution time and increases resource utilization. The research is focused on improving the existing task scheduling strategies to generate the best possible schedules for the inbound tasks. This is achieved by combining the Honey Badger algorithm with the already existing PSO algorithm. The digging and honey-finding phases of the Honey Badger algorithm aid PSO to evade the local optimum and find a superior solution in the available search space. The proposed task scheduling algorithm is implemented and evaluated using the Matlab simulation tool. The simulation results clearly outline that the proposed algorithm in this research is better in terms of reducing the makespan and increasing resource usage compared to several existing meta-heuristic optimization algorithms.

1 Introduction

Cloud computing has gained prominence in recent years as one of the Internet's fastest-growing technologies and an important area of study. The relevance of utilizing the cloud has been made increasingly clear to people and businesses by the expansion of cloud computing recently (Buyya et al. 2017). Users can pay reasonable prices to use the cloud to achieve higher standard services. Additionally, organizations and individuals can pick among 550 cloud platform services offered by over 360 vendors across 22 categories currently. Cloud computing is a technology that is built upon virtualization technology, which makes use of servers with a huge capacity to provide resources to consumers. These

services are either provided using virtual machines or containers which are also known as lightweight VMs.

Moreover, as the popularity of such an "Operational Expenditure" option kept growing among businesses and individuals, the amount of workload in the cloud was increasing simultaneously. Also, several organizations started to acquire huge servers provided by cloud service providers to meet their virtualization demands. Subsequently, to avoid overutilization of a specific virtual machine in the cluster, the concept of load balancing was introduced. Load balancing is one of the crucial components in cloud computing as it helps to divide the workload accurately among several virtual machines by considering a variety of parameters such as resource usage, energy utilization, execution time, and many more. Task scheduling is an important area of research in cloud load balancing. A vast amount of research is completed in this domain where researchers have tried to develop algorithms that make use of multiple factors such as energy consumption and resource utilization to schedule tasks(Arunarani et al. 2019). Several attempts have also been made where the hybridization of meta-heuristic algorithms is implemented to schedule tasks in a VM cluster. However, there are still some gaps in the research already implemented using such hybridization techniques such as longer execution time of tasks and underutilization of resources available. In this paper, a similar hybridization approach of combining Particle swarm optimization and the Honey Badger algorithm is proposed to reduce the makespan and increase the resource utilization of the cluster.

1.1 Research Motivation and Background

The introduction of a unique approach to proactively balance load among virtual machines using a combination of an updated Q-learning and an enhanced Particle Swarm Optimization algorithm was done by (Jena et al. 2022). The objective of maximizing machine productivity by balancing the load among VMs was achieved in this research. Nevertheless, the research was more focused on scheduling the tasks based on priority, which leads to longer waiting times for tasks with lesser weightage. Additionally, a similar attempt at hybridization was made by (Thakur & Goraya 2022) who introduced a brand-new multi-objective optimization algorithmic technique known as PPSO-DA that is based on a dragonfly model and phasor particle swarm optimization. Several attempts were made to integrate the standard particle swarm optimization algorithm with a nature-inspired/meta-heuristic technique. Optimization of the existing PSO algorithm was observed through the results but the issue of the standard PSO algorithm of getting stagnated in finding a better solution persisted across all the existing research. Also, there were no attempts made to integrate the recently proposed Honey Badger algorithm with the standard PSO technique. Hence, this motivated the proposed research to make an attempt of combining the HBA and PSO algorithms to find a better solution in the available search space and overcome the drawback of the standard PSO algorithm.

1.2 Research Objectives

- To implement a hybrid algorithm combining the standard Particle swarm optimization and Honey Badger algorithm.
- To optimize the mathematical formulas for task scheduling to enhance execution time and resource consumption in load balancing.

- To perform a comparative analysis of the proposed system with existing meta-heuristic optimization algorithms.

1.3 Research Question

The research question proposed in this paper is as follows - "To what extent does Hybrid Particle Swarm Optimization Algorithm in task scheduling improves makespan and resource utilization to enhance cloud load balancing."

1.4 Document Structure

This research paper is further divided into 6 sections. Section 2 outlines the related work previously carried out in the domain of task scheduling and hybridization of algorithms. Section 3 describes the research process, including the techniques and algorithms adopted. Section 4 describes the Design specifications of the proposed research. Section 5 documents the step-wise implementation of the proposed load-balancing algorithm. Finally, section 6 and 7 consists of experimental results and conclusion respectively.

2 Related Work

2.1 Energy Aware/Efficient Load balancing techniques.

(Kaur & Aron 2020) explained how workload could only be completed on the fog nodes if they had been assigned to all of the nodes. In this paper, the Energy-Aware Load Balancing algorithm for the fog/edge computing domain was investigated. This algorithm's major objective was to decrease the power needed to utilize Edge resources and to fully utilize those that are available at a fog node. On iFogSim, a comparison between the researched method and the Tabu Search approach was done. The emphasis of the upcoming study would be on applying a meta-heuristic method to adjust the workload in Fog Computing. Next, according to (Velde et al. 2021), the previous development of cloud activities was reorganized and illustrated the scheduling methodology. The general focus of the improvement concerns was Quality of Service in cloud architecture. The "planning technique" at Level-1, which was built on SLA, provided comprehensive information about the jobs and distribution to the nodes. The Level 2 monitoring process for idle machines changed the load on multiple hosts in each group. Instead of making judgments based on ordinary, constrained allotment feedback, the load balancer in the research that is being given must acquire knowledge of the provider's present state. Moreover, (Sohani & Jain 2021) studied the PMHEFT method which is abbreviated as the Predictive Priority-based Modified Heterogeneous Earliest Finish Time algorithm with the aim of determining the application's anticipated resource requirements. The author created a prediction-based method to offer the resource in a heterogamous system efficiently and dynamically while meeting end-user requirements. The tested method assisted in reducing the makespan of a particular series of activities in a virtual machine when the load was evenly spread among all of the VMs. In contrast to other well-known algorithms, the experiment's findings demonstrated that the explored approach was efficient and used less energy.

Furthermore, A dual-staged cloud-based container management system was suggested by (Zhang et al. 2022). In the proposed architecture, stages for deployment and relocation

of the container have been incorporated. In an attempt to distribute the load over the long run, the investigation primarily focused on the container placement method used by cloud nodes. It is characterized as a two-stage technique for container management in the cloud that comprises deployment and relocation. The initial stage was to develop a long-term load-balancing plan for container deployment onto the cloud servers. The second phase was using a rapid container migration strategy among cloud servers. In conclusion, it was advised to use a two-objective optimization approach to lower the load disparity ratio while lowering the relocation expense. Finally, A basic framework for interactive optimization-based scheduling algorithm for Cloud Computing based scenarios was proposed by (Wang et al. 2021). The purpose of this research was to facilitate a model for resource provisioning that supports collaborative optimization of power usage and QoS in a cloud computing environment, hence decreasing data center power consumption while preserving QoS. This research built a multi-VM buffering infrastructure and examined the connections between the queue of job sequencing and the power used by the framework. The results of the studies demonstrated that the scheduling technique has the ability to effectively lower the power needed by cloud data centers while ensuring QoS.

2.2 Leveraging comparative optimization techniques for load balancing

(Haris & Zubair 2021) introduced the Mantaray modified multi-objective Harris hawk optimization a dynamic approach to distributing the workload evenly on the foundation of a computational framework. The Harris Hawk Optimization investigation zone was updated using Manta-Ray Forging Optimization in order to improve price, reaction speed, and equipment use. According to this method's resilience, the performance was raised, the workload remained regulated, as well as harmony among work prioritization was achieved. The results of the simulation (obtained using CloudSim) demonstrated the superiority of the recommended algorithm over competing methods. Additionally, (Balaji et al. 2021) examined the devastating effects on the environment caused by DC's enormous power demand and massive carbon footprint. This study offered a method for increasing the system resource use of the VM while consuming less electricity. With the help of optimal cryptographic encryption techniques, this study presents an approach for relocating virtual machines that are both secure and energy efficient. Using a sanitization method, the sent information is encrypted for safety purposes. The performance is evaluated using parameters such as make span, energy utilization, and memory use. Moreover, A load balancing method enabled by an enhanced Particle swarm scheduling algorithm was described by (Pradhan & Bisoy 2022). This system scheduled tasks using the currently offered cloud environment, thereby shortening the runtime and using more resources. The enhanced task allocation algorithm is based on the Particle Swarm approach, which assesses every particle's optimal solution utilizing the fitness function. This project used the CloudSim modeling toolbox to build the proposed algorithm. The provided approach demonstrated enhanced results in contrast to the other existing strategies with respect to throughput duration and resource consumption in the simulated output.

The two primary difficulties with security and optimal workload balancing in the cloud were described by (Kaviarasan et al. 2022). Monarch butterfly behavior served as a bio-inspiration for the suggested work's structure. The suggested Meta-heuristic method precludes striking a near-optimal solution when looking for the optimal outcome. It is

evident that the exploration speed is higher than singular solution-based methodologies because the technique can migrate into intriguing sections of the investigation area. The findings were compared to many well-known benchmarked algorithms, including Common Scrambling, and Honey Badger algorithm based on experimental research outcomes. The suggested algorithm was shown to have more fault tolerance compared to its competition. Furthermore, It was suggested to solve an optimisation issue using the Artificial Bee Colony load balancing technique (Shen et al. 2019). To enhance the overall effectiveness of the load-balancing approach and obtain enhanced agility, an optimisation strategy of the Artificial Bee Colony is recommended by the author. The ABC approach was improved in this research, and a group of VMs was built using cloud resources from the smart grid. The outcomes demonstrate that the technique may be used to decrease latency, reaction speed, and resource usage in a cloud-based system for a smart grid. The efficiency of the suggested strategy was validated by comparing the outcomes of several simulations. Lastly, (Priya et al. 2019) proposed a task scheduling/load balancing strategy based on resource assignment in the cluster to economically provide cloud services. For the purpose of improving the effectiveness of resource scheduling in the cloud environment, this strategy creates a multifunctional resource scheduling solution that utilizes fuzzy logic. Then, it is feasible to enhance the utilisation of VMs through accurate and logical load balancing by using the Multi-dimensional Queuing Load Optimisation technique to automatically select a query from a group. It is evident from the simulation results that the proposed technique is better in terms of success rate, execution time, and resource scheduling.

2.3 Hybrid algorithms in Load Balancing

(Junaid et al. 2020) suggested a novel hybrid strategy to equalize the information for classifying the quantity of records that exist in the cloud depending upon file type formatting. To evenly spread the burden in the cloud computing environment, a meta-heuristic optimization technique known as Ant Colony Optimization distributed this information dependent on FTF. In respect of SLA violations, the migration duration, performance, delay, as well as optimisation duration, all components of the Quality-of-Service of the suggested system were contrasted to the present approach. Next, (Kruekaew & Kimpan 2022) forecast a MOABCQ-style autonomous job scheduling approach for cloud computing. The Artificial Bee Colony algorithm's output was enhanced in this study using the Multi-Objective Task Scheduling Optimization technique, a supervised learning strategy, and a Q-Learning algorithm. With this approach, planning and asset use were streamlined, VM performance was increased, as well as the workload was distributed across VMs according to computation time, expense, and capacity utilization. The experimental findings showed that the proposed approach worked better to increase productivity and median resource utilisation while minimizing makespan, cost, and degree of imbalance. Moreover, (Li et al. 2020) examined how the cloud computing service providers concentrated upon sharing the computational prospects with the assistance of virtualization. These jobs then were distributed to distant data centers via cloud services to analyze the information. Throughout this period, the scheduling algorithm process was successful in reducing runtime, improving the quality of service, and optimising the overall workload upon that VMs. This approach sought to shorten the number of iterations and increase the level of VM equilibrium.

Furthermore, The Integrated Multi-Objective Particle Swarm Optimization and Fire-

fly techniques were combined to create the upgraded load balancing method proposed in this paper (Devaraj et al. 2020). The IMPSO method and the Firefly algorithm are used in the suggested study to find the heightened responses and also to condense the search space. The recommended FIMPSO solution enhanced critical indicators, such as the best resource usage and task latencies, and provided an optimum overall mean. In order to examine the results, measures such as timeframe, resource utilisation, reliability, makespan, and transmission rate were used. Subsequently, An EDA-GA framework that combines a genetic algorithm with an estimation of distribution approach was proposed by (Pang et al. 2019). Initially, a particular scale of potential solutions was offered using the probabilistic model and sampling method of EDA. Then, crossover extension and mutation operations were included to expand the search field for solutions. In the end, the jobs were distributed among the virtual machines (VMs) using the most effective scheduling methodology. The test findings demonstrated how effectively the created strategy decreased job completion times and enhanced load balancing efficacy. In addition to that, (Thakur & Goraya 2022) introduced a brand-new multi-objective optimisation algorithmic technique known as PPSO-DA that is based on a dragonfly model and phasor particle swarm optimization. As an holistic approach to equalise the workload of active PMs with their anticipated resource capacity, a resource provisioning concept called RAFL is introduced. It is also advised to use PPSO-DA, a hybrid metaheuristic-based optimization methodology, to analyse and effectively use the available search space. This study examined the characteristics of load imbalance between dynamic hardware resources and their estimated resource capacities by implementing simulation testing using CloudSim. Finally, (Jena et al. 2022) introduced a unique approach to proactively balance load among virtual machines using a combination of an updated Q-learning and an enhanced Particle Swarm Optimization, or QMPSO algorithm. The conventional Q-learning weight-based MPSO approach is enhanced in this study to measure the workload of every VM and balance it with the help of objective function. The objective of the proposed algorithmic approach was to maximize machine productivity by balancing load amongst VMs, optimizing VM productive capacity, and keeping a balance among priority workloads through minimization of the workload waiting period.

2.4 Summary of Literature Review and Proposed method

In this literature review, the first section focuses on the papers which have tried to optimize the load-balancing domain using the energy-efficient approach. Using this approach, the existing research managed to improve several quality-of-service parameters in cloud architectures majorly makespan and energy usage. The overall target was on improving the load balancing techniques but focusing mainly on reducing the energy consumption in terms of makespan and relocation. The next set of research papers is based on improving the capability of optimization algorithms in terms of resource allocation. Considerable efforts were made with evident results by several researchers to reduce the overall cost and maximize the equipment utilization while allocating the task to a set of virtual machines. The focus was to improve the Quality-of-Service parameters like throughput, resource scheduling efficiency, and response time by implementing resource allocation optimization techniques beforehand. The last set of papers describes a hybrid architecture approach to balance the load in the cloud. Various researchers took efforts to precisely combine two or more algorithms to improve overall productivity and average resource utilization. The goal of all previous research in this field has been to maximize cluster

utilization and minimize disruption by applying optimization techniques to the clusters upon virtual machine overload. Also, the research conducted in the field of task scheduling utilizing optimization algorithms is based on allocating the task effectively first by taking into account factors like energy consumption or failure rate. Additionally, there have been several attempts to create a hybrid algorithm, but none of the research has concentrated on hybridizing the honey badger algorithm with Particle Swarm Optimization. Therefore, a hybrid particle swarm optimization approach is suggested in this study that combines the benefits of HBA and the standard PSO to determine the best virtual machine to distribute workloads, minimize the makespan, and maximize resource utilization.

3 Methodology

The research proposed in this paper mainly focuses on improving network efficiency while maintaining the standard of Quality-of-Service parameters. The primary objective of this study is to effectively schedule the tasks to the optimal virtual machine available in the cluster depending upon factors like makespan. This goal is achieved by employing a hybrid version of the standard PSO algorithm and comparing it with a set of existing optimization algorithms. The results will be compared based on Total execution time, Resource Utilisation, and Convergence Curve. This section provides a brief overview of methods, procedures, and approaches that have been employed in our study.

3.1 Particle Swarm Optimisation and its drawbacks

PSO, also known as Particle Swarm Optimization, is an organism-inspired algorithm that looks for the best outcome possible inside the available search area. This technique varies from many other optimization strategies because it only needs the fitness method/function and does not rely upon patterns or any kind of distinctive format of the objective.(Juneja & Nagar 2016) The algorithm is highly dependent on the population size hence it is also called as a population-dependent algorithm. It is directly comparable to the genetic algorithm in this respect. In PSO, Particles are indeed a group of discrete components that move through a space in a sequence of phases. The technique analyses the objective function, individually on the particle level at each stage. After such evaluation, the program calculates the updated velocity of every particle. Following a particle's movement, the code is re-evaluated. In a utility computing context, one method of task scheduling workloads is to shift them to an existing virtual machine in the network. This optimization technique/algorithm is mostly used to identify answers for continuous optimization problems without any previous information (Freitas et al. 2020). This leads to numerous task migration and scheduling problems in the infrastructure. Also, sometimes the standard PSO gets stuck in its local optimum and cannot update the global optimum value (gbest). To overcome this drawback of the standard PSO algorithm, hybridization with some other algorithm was required which will help in finding a more efficient solution from the provided search space.

3.2 Hybridisation of PSO with Honey Badger Algorithm

The Honey Badger Algorithm takes its reference points from the animal honey badger, which inhabits deserts and woodlands in Asia and Asian Subcontinent. It uses its sense

of smell and movement to find prey. The HBA algorithm developed implements the behavior of honey badger with the help of two distinct phases. The honey badger first goes through a digging phase during which it relies on its sense of smell to find its prey and the best location to grab it. The next stage is the honey phase, during which the honey badger searches for the beehive by following the honey bird. The HBA algorithm is initialized by calculating the solution inside the lower-bound and upper-bound of the provided search space. For balancing between the exploration and exploitation of the HBA, a density factor alpha is defined (Hashim et al. 2022).

$$\alpha = C * exp^{(-t/T)} \tag{1}$$

Here, T denotes the total number of iterations, t is the current iteration, and C denotes a constant having a value larger than zero. The Initialisation is then followed by the digging phase where the movements of the particle in the solution space are updated using the specific equation. Finally, there is a honey phase where the actual global optimal solution is updated (Hashim et al. 2022).

Given the benefits of the HBA algorithm, A Hybrid algorithm that is defined on a swarm-based approach is proposed in this paper. The amazing foraging skills of honey badgers serve as the inspiration for the proposed algorithm. The HYPSO approach uses a searching method to solve an optimization problem and is given quantitatively. If the PSO algorithm fails to provide with a Global Optimum value, It uses two Honey Badger algorithm strategies, honey finding and digging to offer a successful resolution. In order to find the optimal answer in a bigger landscape region, this HYPSO is supplied with a sufficient population and a variety of procedures.

3.3 Proposed Research Architecture

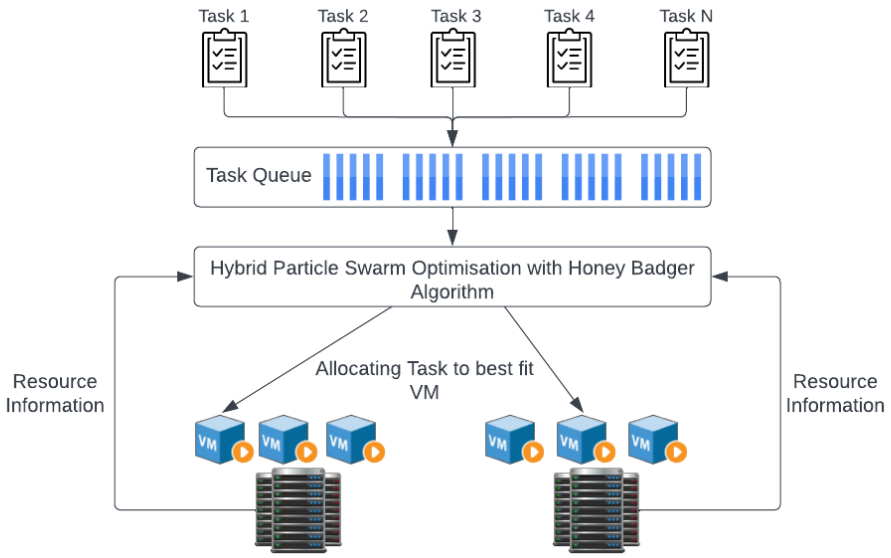

Figure 1: Proposed Research

The workflow is initialized by a user at a remote site by generating a range of workloads that the Virtual Machines will execute, as shown in Figure 2. The spawned set of tasks

are then passed to the task queue. subsequently, the generated series of tasks are then stored in a buffer according to the specified priority. At this stage, in a normal scenario, the tasks will be distributed among the available set of virtual machines using legacy task scheduling algorithms like weighted round robin or least connections. In this case, the current resource utilization of the host VM is not taken into consideration and the tasks are distributed among the available virtual machines. This leads to the problem of overscheduling the tasks on a single VM or in simple words overloads the virtual machine which will lead to a delay in executing the task. To overcome this scenario, in the proposed research (Figure 2) a hybrid particle swarm optimization algorithm is implemented which will calculate the local optimal (pbest) and global optimal (gbest) values using the standard particle swarm optimization algorithm. Additionally, when the standard PSO is stuck in the local optimal and is unable to find a better solution in the search space, the honey badger algorithms digging and honey phases are used to explore more in the search space and update the global optimal value (gbest). In simple words, the Honey Badger algorithm will find the best virtual machine in the available search space to facilitate optimal task scheduling.

3.4 Pseudo-code of the Algorithm

```
Input Data =  P - Population Size, C1 & C2 - PSO algorithms Learning Factors
              W - Inertia Weight, Max_iteration, Flag - HBA Flags to alter direction
              Beta - ability to find optimal best, C - Constant value for HBA
Output Data = Optimal Schedule for Tasks.

set pbest = inf                                    // initialising local optimal as infinite
set gbest = inf                                    // initialising global optimal as infinite

for each particle in P
    Intialise particle                             // Generate random schedule for tasks
Endfor
Do
    for each particle                              // For ever task schedule
        Calculate Fitness function                 // here fitness is considered as makespan
        if Firness(new) > pbest(current)
            update pbest(current) = fitness(new)
        Endif
        if pbest(new) < gbest(current)
            then set gbest(current) = pbest(new)
        else                                       // hybridisation with Honey Badger Algorithm
            calculate α = C*^(-1/max_iterations)
            Calculate the intensity using D and S  // here D is distance and S is concentration strength
            for i = 1:N                            // N is No. of Tasks
                Update the position using HBA digging phase
            Endfor
        EndIf
    Endfor

    for each particle in P
        Calculate particle velocity using PSO
        Set position = velocity + position
    Endfor
While Max iteration reached or minimum error criteria is not attended
```

Figure 2: Pseudocode of the proposed algorithm

The proposed algorithm aims to successfully schedule incoming tasks while maintaining the Quality of Service parameters. The suggested approach uses a Hybrid particle swarm optimization technique to increase the network efficiency. At the initial stage, the algorithm captures parameters such as weight (w), Learning factors (c1 & c2), population size (p), Flag (F), and maximum iterations which are required by the Standard PSO and Honey badger algorithm. Next, it initializes the local and global best values

as infinite so that the fitness function of the proposed algorithm can update these values in the first iteration. Next, for every particle in the population, that is for every task, a random schedule is generated initially. Further, a loop is executed which will run until it reaches the maximum number of iterations defined, which in our case is 100, or until the minimum error condition is not met. Then, for every particle in the population, a fitness/objective function is calculated and compared with the existing pbest value. If the new fitness generated is better than the existing value of pbest, then the value of the newly generated fitness is assigned to the pbest variable. Next, a similar comparison between the pbest and gbest values is performed and if the current gbest value is greater than the pbest, the value of pbest is assigned to the gbest variable. In contrast, if the value of pbest is not smaller than gbest, then the calculation of the density factor alpha and Intensity is carried out to initiate the Honey Badger algorithm. Subsequently, a for loop is executed which will start the execution of the digging and honey-finding phase of the HBA algorithm which will aid PSO to find a better solution to schedule that task in the available cluster of virtual machines. After that, for every particle in the population, that is for every task a velocity is calculated. In the proposed algorithm, Velocity is the information exchanged between the task and the virtual machine. Lastly, the position is updated using the value of velocity, and optimal schedules for tasks are generated. In addition to that, performance factors such as Makespan, Convergence Curve, and Resource utilization are also recorded for evaluation.

3.5 Evaluation Parameters

For the purposes of this study, relevant parameters were determined for algorithms evaluation and to perform a comparison with an existing task scheduling method. The primary objective of the algorithm is to efficiently schedule the incoming tasks, calculate the time consumed for execution (Makespan) and determine the resource utilization. Also, the convergence curve is an important factor when it comes to optimization-based task scheduling algorithms which is also taken into account while comparing the proposed research with the existing ones. The performance measurement factors were selected such that the algorithm's effectiveness could be easily evaluated. The three parameters used for the evaluation are as follows.

- **Makespan -** This is the overall amount of time consumed for the execution of the tasks after scheduling. The time will be reduced as the proposed algorithm will distribute every task to an optimal virtual machine which will be the best fit to perform its execution.

- **Resource Utilisation -** This parameter defines the total amount of resources in the cluster utilized for the execution of allocated tasks. Resource utilization will also be optimized as the proposed algorithms' fitness function takes into consideration the load of every virtual machine and schedules the task to the least loaded one in the network. This maintains a balance in resource utilization and also protects the infrastructure from failure due to overloading.

- **Convergence Curve -** This parameter describes the overall number of iterations which is required by the algorithm to produce an optimal solution. In our case, the convergence curve of the Hybrid particle Swarm Optimisation algorithm will be evaluated to determine the speed of finding the optimal solution.

4 Design Specification

A significant problem in load balancing is the algorithm's capability to accurately assign activities to a collection of virtual machines in such a way that the entire cluster maintains a Highly Significant Quality of Service. Also, the existing PSO algorithm has significant drawbacks which are required to be addressed. The standard PSO algorithm gets stagnated in finding a local optimal in the predefined search space, which in our case is the cluster of VM. Hence, to facilitate a Load Balancing Algorithm that overcomes these drawbacks and provides Improved Quality of Service, we developed a hybrid algorithm that improves on the one developed by (Pradhan & Bisoy 2022).

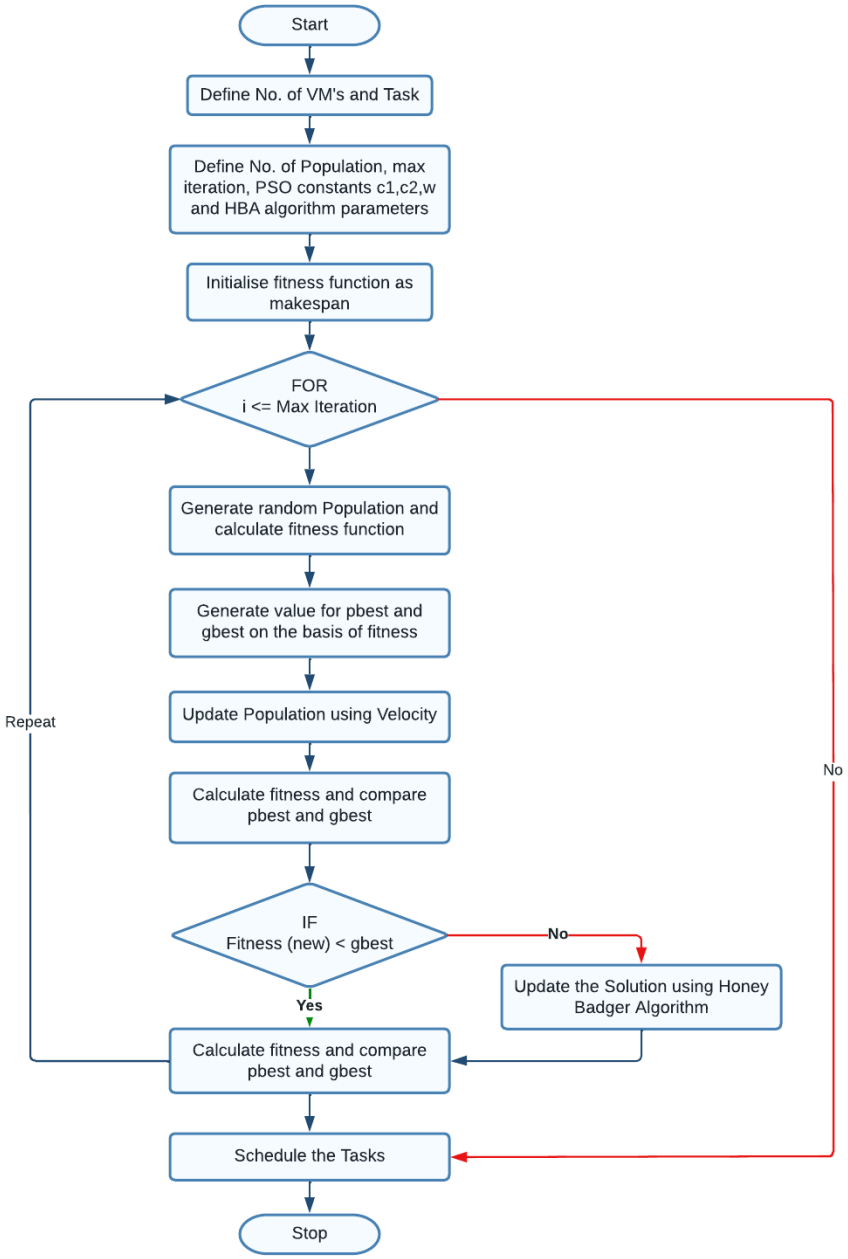

Figure 3: Flowchart of the proposed HYPSO algorithm

The proposed research integrates the HBA algorithm with Standard PSO to maximize resource utilization and makespan, whereas preceding research was focused on optimizing the Standard PSO by taking into consideration factors like task and resource information. Figure 3 represents the flowchart of the proposed HYPSO algorithm. The Process starts with Defining the number of cloudlets(VMs) and Tasks to be assigned in the cluster. Next, Several parameters and constant values are initialized which are required by the Standard PSO and HBA algorithm to be implemented such as weight, maximum iterations, and intensity. After that, makespan is initialized as the fitness function which is also called the objective function in terms of PSO, and the For loop is executed which repeats the process until it reaches the maximum number of iterations. When the loop executes for the first time, it generates a random population to calculate the fitness function which is based on makespan, and also sets the value for pbest (Local Optimum) and gbest(Global optimum) depending on the value generated by the fitness function. Further, it updates the population generated using velocity and calculates the fitness to compare the pbest and gbest values generated. If the value generated by the fitness function is less than the gbest value, it updates the gbest value with the new fitness value. Also, if the newly generated fitness value is not less than the current value of gbest, the Honey Badger algorithm is initialized to find the best solution and overcome the drawback of Standard PSO. After that, once again the fitness value is calculated and compared with pbest. If the newly generated solution is better than the one generated previously, the global value is updated and the loop is repeated. The loop keeps on executing until the maximum number of iterations is completed and all the scheduled tasks are allocated to the most efficient virtual machine.

5 Implementation

Given the expense of building up a real-world cloud computing infrastructure, the research has chosen to develop the suggested hybrid method using simulations. In this case, MATLAB is used to illustrate and carry out the complex computations necessary for an algorithm based on optimization techniques. Prior to the live deployment of the suggested method, using a simulator will aid in detecting the majority of errors. The benefit of employing this configuration is that we may take preventative action, saving us the expense of setting up a live testing environment. Additionally, The Simulation software was installed on a device running a 64-bit Windows 11 operating system with an AMD Ryzen 5, 6-core CPU. Moreover, the built-in IDE offered by MATLAB was utilized in conjunction with the C language to develop the suggested optimization method.

5.1 Tool used for Evaluation

MATLAB, a tool provided by MathWorks, is a specialized programming interface for developing techniques to replicate the operation of algorithms. It discusses problems and produces solutions by merging technology, and graphic elements, with programming in an easy-to-use user interface, all while employing a properly ordered technical terminology. The unified simulation model, MATLAB considers a collection to constitute a simple data object if that array does not require it to be dimensioned. This allows many advanced programming tasks, particularly those that use vector or matrix compositions, to be completed much more quickly than they could be by implementing programming languages like C. (CIMMSEducation 2020)

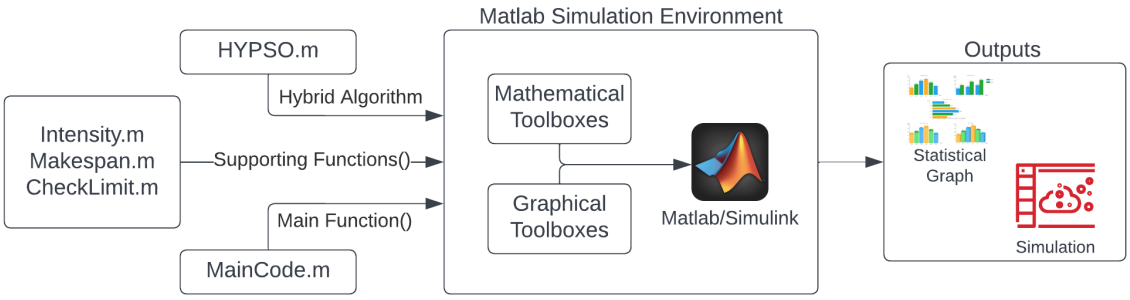

Figure 4: Architecture of Matlab Implemented

Additionally, it permits us to explore with a wide range of ideas or methods. By adopting a high-level coding interface, which takes care of lower-level programming concerns like resource use and parameter class effectively, users may focus on how the algorithm must operate. Once it has been operationally tested, the load balancing technique could be enhanced for effectiveness and dependability. Tools included within the system can detect problems and provide fixes. To ensure that the approach works consistently on predefined CPUs, we can employ a collection of data structures and arithmetic calculations (Mathworks 2018). To implement the proposed research, We have utilized several in-built toolboxes provided by Matlab.

5.2 Formulating the Objective function required

The Particle Swarm optimization algorithm requires an objective function to run in the background to calculate the fitness of a particular task. Makespan is considered as the objective function in our case, to minimize the overall execution time of the tasks. Also, this objective function acts as a main pillar for the newly proposed HYPSO algorithm. To define this characteristic of the optimization algorithm, which is to calculate the fitness function, we have implemented a Matlab function and stored it in a separate file called "fmakespan.m".

fmakespan.m

```
function fit= fmakespan(pSol,nProcess)
No=size(nProcess,1);
T=zeros(1,No);
while 1
  for i=1:No
    T(i)=T(i)+nProcess(i,pSol(1));
    pSol(1)=[];
    if isempty(pSol)
        break;
    end
  end
   if isempty(pSol)
        break;
    end
end
fit=max(T);
end
```

Figure 5: Objective Function to calculate fitness

This function is then utilized in the "MainCode.m" file to calculate the objective function (Figure 5). Also, to make it simple to call this function in the code, we have

created a function handle using the "@" symbol which is reserved for function handling in Matlab. Furthermore, the newly created function handle also acts as the main argument to be passed for the execution of the proposed Hybrid Particle Swarm Optimisation algorithm.

5.3 Checking the Upper bounds of position generated

To check the upper bounds of the solution generated for the position of the particle in the solution space, we have formulated a Matlab function called "CheckLimit.m" (Figure 6). In simple terms, the function checks the solution generated by the Hybrid PSO algorithm, that is the position (Virtual Machine) to which the task will be assigned. Also, it checks whether the position of the particle is inside the upper limit of the virtual machine cluster.

fmakespan.m | MainCode.m | CheckLimit.m

```
function [bSolution] = CheckLimit(bSolution,ULlimit,dim)
bSolution=round(bSolution);
for i=1:size(bSolution,1)
    for j=1:size(bSolution,2)
        if bSolution(i,j)<=ULlimit(1)
            bSolution(i,j)=randi(ULlimit,1);
        end
        if bSolution(i,j)>=ULlimit(2)
            bSolution(i,j)=randi(ULlimit,1);
        end
        if bSolution(i,j)<0
           bSolution(i,j)=randi(ULlimit,1);
        end
    end
    if length(unique(bSolution(i,:)))~=dim
        temp=randperm(ULlimit(2));
        bSolution(i,:)=temp(1:dim);
    end
end
```

Figure 6: The CheckLimit.m function

This function is created and stored in a separate Matlab file for the purpose of code re-usability. The Defined function first rounds up the value of solution-generated and checks for four conditions. First, it checks whether the solution generated for the task placement is under the upper-bound defined or not. If it is not under the defined limit, it generates a random value between the upper bound defined. Furthermore, it continues to check conditions like "IF the value is negative" or "IF the value is Equal to Zero after roundup" and performs a similar procedure of generating a random value between the specified upper bounds as defined in the first "IF" condition. Lastly, The function also checks whether the position is unique in every iteration to avoid virtual machine overloading which can lead to failure in task execution.

5.4 Calculating the Intensity Factor HBA

As the proposed algorithm is a combination of Particle Swarm Optimisation and the Honey Badger algorithm, calculating the intensity is an important aspect of the proposed research. The Intensity factor is derived from the "strength of the prey" (defined as S) and the distance between the "prey and the ith honey badger" (defined as D).

To define this functionality in Matlab, we have designed a function called "Intensity.m" (Figure 7) and stored it in an isolated ".m" file for the purpose of code reuse.

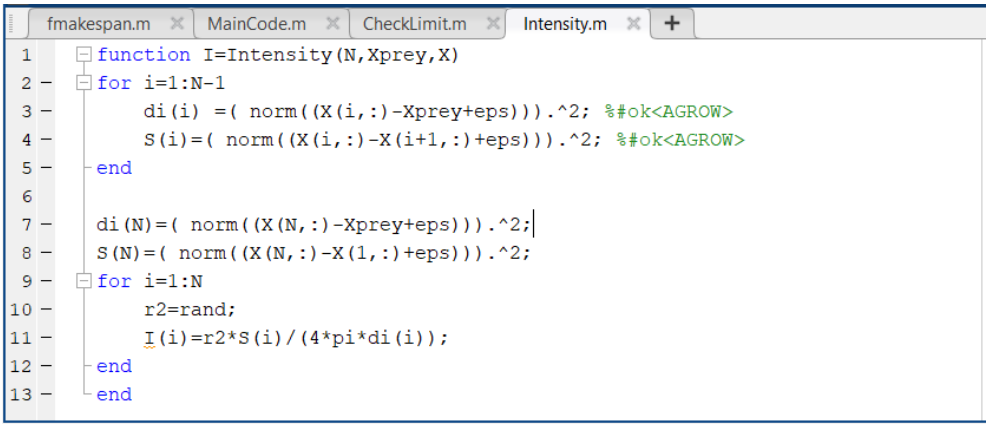

Figure 7: The Intensity.m function

Also, while calculating the values for S and D in the Matlab function, we have used an "eps" value which is by default defined in Matlab. The "eps" value is a very small number (for instance 2.2204e-16) which avoids the multiplication value generated from being zero. This value is applied here as the formula for calculating the Intensity has a divisible value of the parameter "D" and multiplication value of parameter "S".

5.5 Hybridisation of PSO and HBA

To simulate the working of the proposed algorithm, we have created an "HYPSO.m" file (Figure 8) which consists of programmatical steps to execute the Hybrid algorithm. This function consists of all phases of the Particle swarm optimization algorithm and some phases of the Honey Badger algorithm. This is to overcome the drawback of stagnation in the standard PSO algorithm of finding the local optimum. The recently proposed HBA algorithm helps PSO to better explore the available search space to find a better "Global Optimum Value".

Figure 8: Integration of PSO and HBA through Matlab

The Function starts with finding the solution by using the PSO algorithm. It updates the Local Optimal(pbest) and Global Optimal(gbest) values and also compares them with previous iterations. Additionally, when the PSO algorithm gets stuck and is unable to find the best virtual machine for task allocation, starting from the "Initialisation phase" to the "Digging phase" of the Honey Badger Algorithm are executed and the position value is updated with a better alternative. Finally, using the updated values, the velocity of the particles and position is calculated and an optimal schedule for tasks is generated.

6 Evaluation

In this section, a comprehensive evaluation is performed to evaluate the proposed Hybrid Particle Swarm Optimisation algorithm. The algorithm is implemented using Matlab and simulation is performed to generate results. The proposed algorithm is then compared using several factors such as Convergence Curve, Execution time, and Resource utilization. The proposed task-scheduling algorithm is a combination of optimization techniques called Particle Swarm Optimisation and Honey Badger Algorithm which are then compared with several existing optimization techniques documented in (Pradhan & Bisoy 2022).

The experiments were conducted by generating a specific amount of workload that is to be distributed among a fixed number of virtual machines. Using Matlab, we have simulated this exact scenario where a specified number of tasks are created by assigning them a makespan time. Makespan in our case is a random number that is generated using Matlab and assigned to the task which will indicate the exact amount of time required by the task to complete its execution. To evaluate the results dynamically, three experiments/cases were considered where factors such as the number of Virtual machines in the cluster and the required amount of tasks to be assigned were varied while keeping other factors constant.

Table 1: Experiment Setup

Experiment/ Scenario	Cluster Size	Total No. of Tasks
1	3 VM's	10
2	5 VM's	15
3	15 VM's	25

6.1 Experiment / Case Study 1

In this experiment, we are going to perform tests on the proposed hybrid PSO algorithm and analyze the test results by using QoS parameters. We are going to compare and analyze the performance of the algorithm by considering factors like Resource Utilisation and Makespan. Furthermore, an in-depth comparison of the results with already existing meta-heuristic optimization techniques like LBMPSO, PSOBTS, L-PSO, TBSLB-PSO, and DLBA is performed.

As mentioned in Table 1, we are considering scenario 1, where we have a cluster of 3 Virtual machines to assign 10 tasks with 30 processes. The results from the tests are mentioned in Figures 9, 10, and 11.

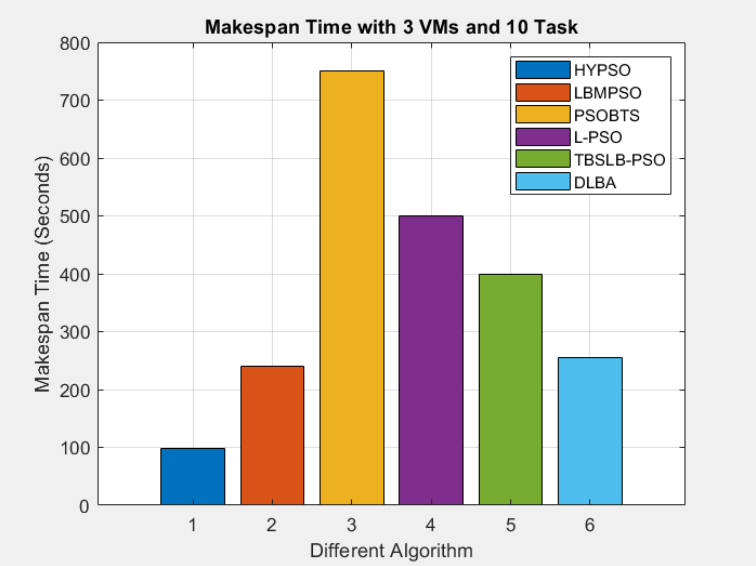

Figure 9: Makespan of HYPSO with 3 VMs and 10 Tasks

Results of makespan for a cluster size of 3 Virtual machines and 10 tasks are depicted in figure 9. From the results, we can observe that our goal to reduce the makespan of the tasks is achieved successfully. Also, when we compare the results with other meta-heuristic techniques, the proposed HYPSO algorithm has a very minimal makespan compared to the existing techniques.

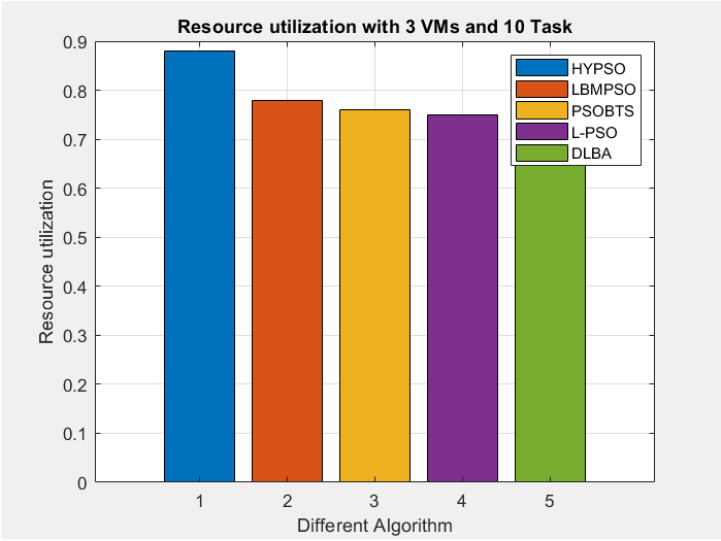

Figure 10: Resource Utilisation of HYPSO with 3 VMs and 10 Tasks

Results of resource utilization for a cluster size of 3 Virtual machines and 10 tasks are outlined in figure 10. From the results, it is evident that the tasks are distributed evenly among all the virtual machines in the cluster, and hence increment in resource utilization is observed. Also, the best optimal solution was found on the 40th iteration as observed from figure 11. The convergence curve is a parameter that defines how quickly the algorithm completes the optimization process and finds the best optimal solution for task scheduling.

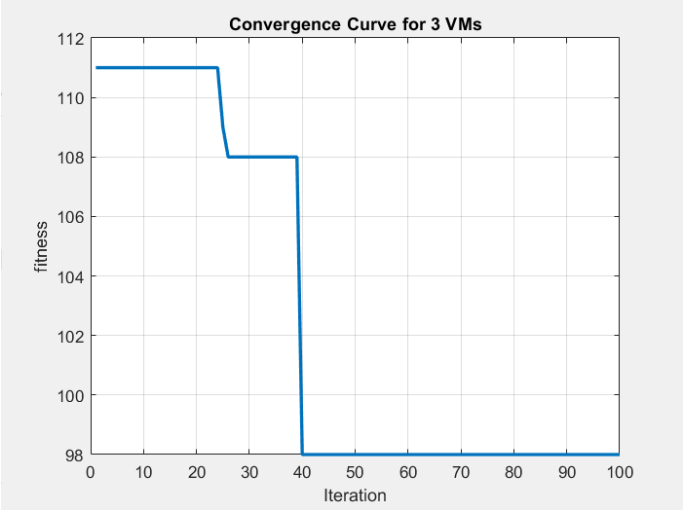

Figure 11: Convergence Curve of HYPSO with 3 VMs and 10 Tasks

6.2 Experiment / Case Study 2

In the second experiment, We tried to change the configuration of the Virtual machine cluster and the number of tasks assigned to them. Analysis of the results obtained from this experiment will be determined by factors like resource utilization and execution time required to complete the tasks. Comparison with already existing meta-heuristic algorithms will be made to evaluate the performance improvement. As mentioned in Table 1, we are considering scenario 2, where we have a cluster of 5 Virtual machines to assign 15 tasks with 30 processes. The results from the tests are mentioned below in Figures 12, 13, and 14.

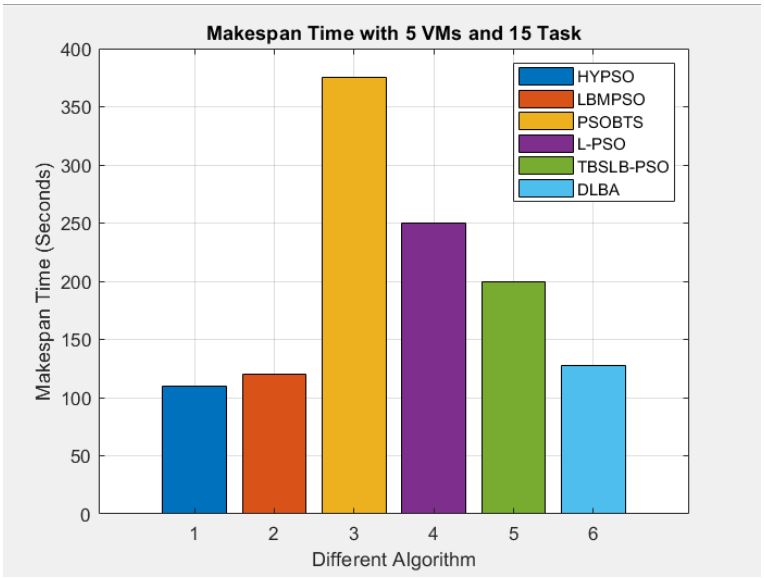

Figure 12: Makespan of HYPSO with 5 VMs and 15 Tasks

Results of makespan for a cluster size of 5 Virtual machines and 15 tasks are mentioned in figure 12. From the results, it can be noted that the proposed Hybrid PSO algorithm is slightly better than the LBMPSO algorithm but outperforms others marginally.

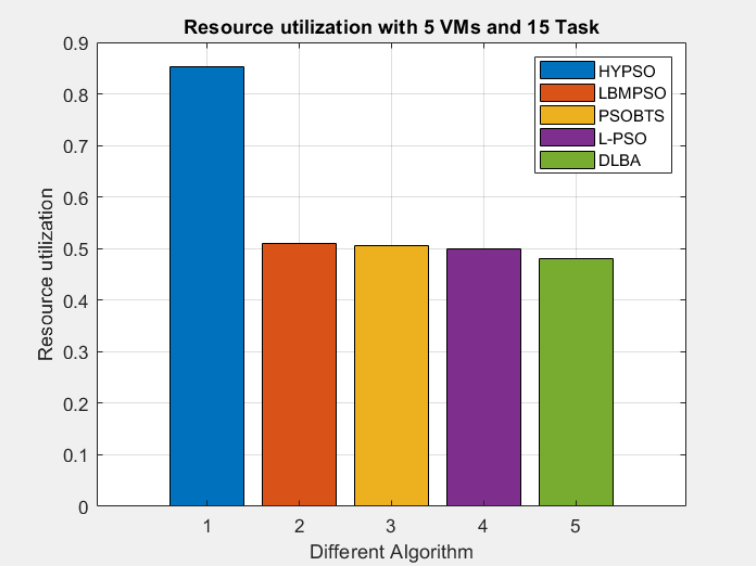

Figure 13: Resource Utilisation of HYPSO with 5 VMs and 15 Tasks

Results of resource utilization for a cluster size of 5 Virtual machines and 15 tasks are mentioned in figure 13. From the results, it is evident that the tasks are distributed evenly among all the virtual machines in the cluster, and hence increment in resource utilization is observed. Also, the resource utilization of the cluster is almost 85% which directly reflects improvements in cluster efficiency and utilization when applied in a practical scenario. Additionally, the optimal solution was found on the 28th iteration as observed from the Figure 14

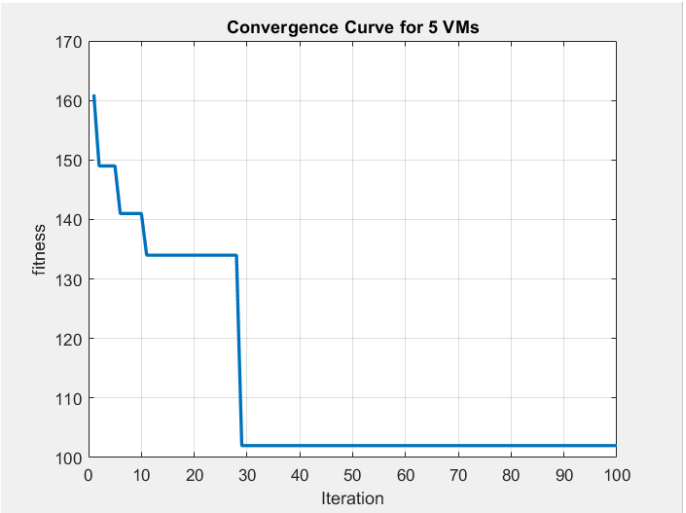

Figure 14: Resource Utilisation of HYPSO with 5 VMs and 15 Tasks

6.3 Experiment / Case Study 3

In the third experiment, we tested the proposed algorithm by altering the virtual machine cluster's setup and the amount of workload that was given to the cluster. As mentioned in Table 1, we are considering scenario 3, where we have a cluster of 15 Virtual machines to assign 28 tasks with 30 processes. Resource utilization and the amount of time needed

to execute the activities were key aspects in the analysis of the outcomes from this experiment. To assess the performance increase, comparisons with already existing meta-heuristic algorithms were performed. The results from the tests are mentioned below in Figure 15.

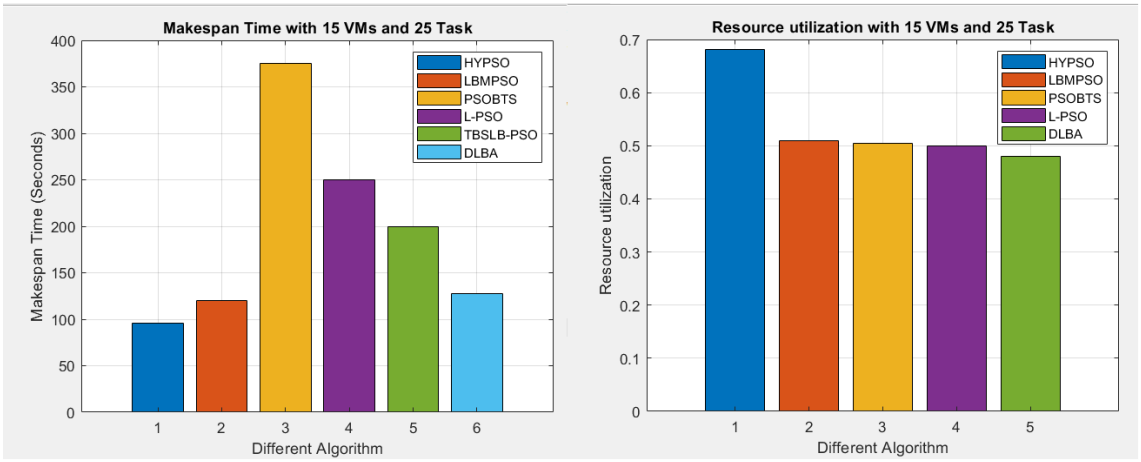

Figure 15: Makespan of HYPSO with 15 VM's and 25 Tasks

Results of makespan for a cluster size of 15 Virtual machines and 25 tasks are mentioned in figure 15. From the results, It is evident that the proposed Hybrid PSO algorithm can also effectively reduce the makespan in a larger cluster of virtual machines. Also, when compared to a few existing meta-heuristic algorithms, the proposed approach outperforms every algorithm marginally. Figure 15 mentions the results of resource utilization for a cluster size of 15 Virtual machines and 25 tasks. Better task distribution throughout the virtual machines in the cluster is apparent from the data, and a rise in resource usage is therefore observed. Additionally, the cluster uses approximately 70% of the available resources in the cluster while other algorithms use less than 50%.

6.4 Discussion

Based on the results of the above experiment, we can claim that the Hybrid load balancing algorithm in cloud computing is effective at task scheduling based on virtual machine capacity and fitness function. It even reveals that our suggested technique would prevent the virtual machines in the cluster from being overloaded. Also, the experiments prove that the suggested approach can withstand and be deployed on clusters of varied sizes and will still give better results than any other existing optimization approaches. Furthermore, the Hybrid PSO algorithm also distributed the tasks evenly so that the maximum amount of resources available in the cluster would be utilized to avoid the problem of underutilization of resources in the cluster.

Moreover, as the world is getting more aware of green computing and energy efficiency, increased resource utilization can lead to a higher amount of energy consumption. However, In a practical situation, the suggested algorithm will succeed in producing better outcomes. It can also produce better results when implemented in a bigger cluster because the availability of resources to schedule the task increases marginally.

7 Conclusion and Future Work

Due to the rising demand for cloud computing services and applications, load balancing is turning into one of the most critical piece of technology for distributing workloads across the virtual machines available in a cluster. Nevertheless, when it comes to task scheduling, there are substantial challenges in the load-balancing domain. To address these issues, we conducted research to reduce makespan and improve resource utilization in the area of task scheduling. To generate the best schedule possible for the inbound tasks, we suggested combining two meta-heuristic algorithms through a hybridization process. While scheduling the tasks to a cluster of virtual machines, the suggested approach in this study successfully minimized the execution time when compared to various other existing meta-heuristic algorithms. Additionally, this research also achieved the aim of maximal resource utilization and reduced idle time of VMs in the cluster. The results are evident and can be verified through the simulation outputs produced by Matlab.

However, in terms of limitations and future work, the proposed task scheduling algorithm is designed to utilize the cluster optimally and reduce the execution time of tasks. This in turn is increasing the amount of resource usage at any point in time which leads to a rise in overall energy consumption. Hence, there is scope to conduct additional research in making the proposed algorithm more energy aware and put forward a more greener solution.

References

Arunarani, A., Manjula, D. & Sugumaran, V. (2019), 'Task scheduling techniques in cloud computing: A literature survey', *Future Generation Computer Systems* **91**, 407–415.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X17321519*

Balaji, K., Kiran, P. S. & Kumar, M. S. (2021), 'An energy efficient load balancing on cloud computing using adaptive cat swarm optimization', *Materials Today: Proceedings* .
URL: *https://www.sciencedirect.com/science/article/pii/S2214785320387125*

Buyya, R., Srirama, S. N., Casale, G., Calheiros, R. N. & et al., Y. S. (2017), 'A manifesto for future generation cloud computing: Research directions for the next decade', *CoRR* **abs/1711.09123**.
URL: *http://arxiv.org/abs/1711.09123*

CIMMSEducation (2020), 'Introduction to matlab and its workfolow'.
URL: *https://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm*

Devaraj, A. F. S., Elhoseny, M., Dhanasekaran, S., Lydia, E. L. & Shankar, K. (2020), 'Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments', *Journal of Parallel and Distributed Computing* **142**, 36–45.
URL: *https://www.sciencedirect.com/science/article/pii/S0743731520300459*

Freitas, D., Lopes, L. G. & Morgado-Dias, F. (2020), 'Particle swarm optimisation: A historical review up to the current developments', *Entropy* **22**(3).
URL: *https://www.mdpi.com/1099-4300/22/3/362*

Haris, M. & Zubair, S. (2021), 'Mantaray modified multi-objective harris hawk optimization algorithm expedites optimal load balancing in cloud computing', *Journal of King Saud University - Computer and Information Sciences* .
URL: *https://www.sciencedirect.com/science/article/pii/S1319157821003372*

Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S. & Al-Atabany, W. (2022), 'Honey badger algorithm: New metaheuristic algorithm for solving optimization problems', *Mathematics and Computers in Simulation* **192**, 84–110.
URL: *https://www.sciencedirect.com/science/article/pii/S0378475421002901*

Jena, U., Das, P. & Kabat, M. (2022), 'Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment', *Journal of King Saud University - Computer and Information Sciences* **34**(6, Part A), 2332–2342.
URL: *https://www.sciencedirect.com/science/article/pii/S1319157819309267*

Junaid, M., Sohail, A., Ahmed, A., Baz, A., Khan, I. A. & Alhakami, H. (2020), 'A hybrid model for load balancing in cloud using file type formatting', *IEEE Access* **8**, 118135–118155.
URL: *https://ieeexplore.ieee.org/abstract/document/9121263*

Juneja, M. & Nagar, S. K. (2016), Particle swarm optimization algorithm and its parameters: A review, *in* '2016 International Conference on Control, Computing, Communication and Materials (ICCCCM)', pp. 1–5.
URL: *https://ieeexplore.ieee.org/abstract/document/7918233*

Kaur, M. & Aron, R. (2020), 'Withdrawn: Energy-aware load balancing in fog cloud computing', *Materials Today: Proceedings* .
URL: *https://www.sciencedirect.com/science/article/pii/S2214785320387277*

Kaviarasan, R., Harikrishna, P. & Arulmurugan, A. (2022), 'Load balancing in cloud environment using enhanced migration and adjustment operator based monarch butterfly optimization', *Advances in Engineering Software* **169**, 103128.
URL: *https://www.sciencedirect.com/science/article/pii/S0965997822000394*

Kruekaew, B. & Kimpan, W. (2022), 'Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning', *IEEE Access* **10**, 17803–17818.
URL: *https://ieeexplore.ieee.org/abstract/document/9708723*

Li, W., Tang, Z. & Qi, F. (2020), A hybrid task scheduling algorithm combining symbiotic organisms search with fuzzy logic in cloud computing, *in* '2020 IEEE 23rd International Conference on Computational Science and Engineering (CSE)', pp. 16–23.
URL: *https://ieeexplore.ieee.org/abstract/document/9345878*

Mathworks (2018), 'Algorithm development and simulink solutions'.
URL: *https://www.mathworks.com/solutions/algorithm-development.html*

Pang, S., Li, W., He, H., Shan, Z. & Wang, X. (2019), 'An eda-ga hybrid algorithm for multi-objective task scheduling in cloud computing', *IEEE Access* **7**, 146379–146389.
URL: *https://ieeexplore.ieee.org/abstract/document/8862833*

Pradhan, A. & Bisoy, S. K. (2022), 'A novel load balancing technique for cloud computing platform based on pso', *Journal of King Saud University - Computer and Information Sciences* **34**(7), 3988–3995.
URL: *https://www.sciencedirect.com/science/article/pii/S1319157820304961*

Priya, V., Sathiya Kumar, C. & Kannan, R. (2019), 'Resource scheduling algorithm with load balancing for cloud service provisioning', *Applied Soft Computing* **76**, 416–424.
URL: *https://www.sciencedirect.com/science/article/pii/S1568494618307105*

Shen, L., Li, J., Wu, Y., Tang, Z. & Wang, Y. (2019), Optimization of artificial bee colony algorithm based load balancing in smart grid cloud, *in* '2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)', pp. 1131–1134.
URL: *https://ieeexplore.ieee.org/abstract/document/8881232*

Sohani, M. & Jain, S. C. (2021), 'A predictive priority-based dynamic resource provisioning scheme with load balancing in heterogeneous cloud computing', *IEEE Access* **9**, 62653–62664.
URL: *https://ieeexplore.ieee.org/abstract/document/9410259*

Thakur, A. & Goraya, M. S. (2022), 'Rafl: A hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment', *Simulation Modelling Practice and Theory* **116**, 102485.
URL: *https://www.sciencedirect.com/science/article/pii/S1569190X21001702*

Velde, V., Enumala, K. & Bandi, K. (2021), 'Optimized adaptive load balancing algorithm in cloud computing', *Materials Today: Proceedings* .
URL: *https://www.sciencedirect.com/science/article/pii/S2214785321008476*

Wang, B., Liu, F., Lin, W., Ma, Z. & Xu, D. (2021), 'Energy-efficient collaborative optimization for vm scheduling in cloud computing', *Computer Networks* **201**, 108565.
URL: *https://www.sciencedirect.com/science/article/pii/S1389128621004783*

Zhang, W., Chen, L., Luo, J. & Liu, J. (2022), 'A two-stage container management in the cloud for optimizing the load balancing and migration cost', *Future Generation Computer Systems* **135**, 303–314.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X22001674*