



An Adaptive Neuro-Fuzzy Inference System and Black Widow Optimization Approach for Optimal Resource Utilization and Task Scheduling in a Cloud Environment

Manikandan Nanjappan¹ · Gobalakrishnan Natesan² · Pradeep Krishnadoss³

Accepted: 19 July 2021 / Published online: 10 August 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

With the enhancing demand of the cloud computing products, task scheduling issue has become the hot study topic in this area. The task scheduling issue of the cloud computing method is more difficult than the conventional distributed system. The majority of the previous scheduling schemes use virtual machine (VM) instances, which takes enormous start up time and requires the full resources to perform the tasks. The proposed approach utilizes an Adaptive Neuro-Fuzzy Inference System (ANFIS)-Black Widow Optimization (BWO) (ANFIS-BWO) method for establishing the proper VM for every task with less delay. Resource scheduling is another important objective for optimal usage of resources (servers) in the cloud environment. The BWO algorithm is used to obtain the best solution in the ANFIS scheme. The proposed approach can employ the VMs on the best server by the optimal scheduling scheme. The main aim of the proposed approach is to minimize the computational time, computational cost, and energy consumptions of the tasks with useful resource utilization. We describe that the proposed approach performs better than the existing approach concerning performance metrics such as computational time, makespan, energy consumption, computational cost, and resource utilization.

Keywords Cloud computing · Resource utilization · Task scheduling · Makespan · Virtual machine · BWO · Computational time · Energy consumption

✉ Manikandan Nanjappan
macs2005ciet@gmail.com

¹ Department of Computer Science and Engineering, St. Joseph's College of Engineering, Chennai, Tamilnadu, India

² Department of Information Technology, Sri Venkateswara College of Engineering, Chennai, Tamilnadu, India

³ School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, Tamilnadu, India

1 Introduction

The Internet of Things (IoT) is the system of physical objects, vehicles, buildings, pieces of equipment, and other items that are enclosed by electronics, sensors, network connectivity, and software, allowing these objects to collect and interchange data. IoT consists of three major categories: objects, transmission networks, and systems utilizing streaming data from and to the objects. Mobile Cloud Computing (MCC) combines numerous technologies for increasing capacity and also existing framework performance. MCC is defined as the combination of cloud computing expertise and mobile equipment to produce resources concerning computational power, energy, context awareness, memory, and storage [1]. There are four categories of cloud computing consumption models. They are private cloud, hybrid cloud, community cloud, and public cloud. It also consists of various service models such as software-as-a-service (SAAS), infrastructure-as-a-service (IAAS) and platform-as-a-service (PAAS) [2]. Cloud computing permits entry to mutual system resources resultant in high computing power with reasonably little management endeavors. Virtual machines (VM's) can be influenced to use the physical machine resources devoid of directly cooperating with it. The conventional cloud system is hugely popular; it has elevated starting setup and upholding cost. In a network-disconnected area [3], the traditional cloud fails to perform any function that involves diffusion of the sensor or measurement.

Because multiple IoT devices are grouped, the amount of data to be processed in the cloud sector has been substantially increased. The cloud is controlled in a way that all the client resources and the server responds to the data demanded by the exterior clients via the storage and server's datasets [4]. This computing can be used broadly in numerous fields. But there are several issues that have been appeared in the IoT development. Cloud computing has difficulty in position-aware applications and is rewardingly delay-sensitive, but can hardly be installed in large quantity due to the high construction cost in front of massive IoT devices [5]. The improved Scheduling Algorithm for QoS is moderately quicker than the target obsessed scheduling and consequently, it is appropriate for more effectually saving and recovering E-health credentials from the cloud. The novel task scheduling algorithm issued to enhance the QoS attributes that consist of metrics like computation time, response time, cost, and availability. [6].

Dynamic Voltage/Frequency Scaling (DVFS)-enabled Energy-efficient Workflow Task scheduling algorithm (DEWTS) can influence the valuable slack time frequently after servers are combined. DEWTS initially computes the original scheduling arrangement for all the tasks and achieves the entire makespan and DEWTS firstly calculates the initial scheduling order of all tasks and obtains the whole makespan and deadline depends on the Heterogeneous-Earliest-Finish-Time (HEFT) algorithm. The tasks can be disseminated in the unused slots beneath the lesser voltage and frequency utilizing the DVFS method and in the task slacking stage the relaxed makespan is enhanced without breaking the dependence restrictions [7]. The container system introduced in 2013 was soon transformed by the liberating Docker project. This is one of the virtualization categories and is very light when compared to conventional virtualization. Due to this benefit, the containers have been approved in the cloud system vigorously. General-Purpose Scheduling structure represents helpful resource organization roles to cloud suppliers or researchers [8].

In cloud computing, the conventional technique utilizes the virtual machine (VM) that offers hardware virtualization in cloud computing. The VM's can work on the physical machine top utilizing a hypervisor and this hypervisor works on the host machine. In the

virtualization cloud, the energy consumption is decreased in real-time hence the total energy observed by the hypervisor working on the host machine top is fairly high and has not been understood until now. For the allocation of resources, the containerized cloud and theoretic approach of the game are to decrease the net energy consumption. The game-theoretic approach is used in both containerized cloud and VM based cloud and this containerized cloud absorbs less energy when compared with VM based cloud [9].

Due to its performance of near-native and small virtualization overhead, the container-based virtualization has newly fascinated more concentration. The mobile cloud is constructed by employing this type of virtualization is a talented thought, although it has numerous difficulties. At the starting stage, the container works on a similar OS as the host. A lot of effort is required to modify the container to maintain mobile applications. In the second stage, OS containers present fewer separation among mobile operating systems (OSes), making it probable to exploit optimizations among mobile applications to enhance the additional performance. Conversely, the optimization is guided by the complete study of offloading performance [10]. Containers are the building blocks of OS-levels virtualization that allows isolated virtual environments without hypervisors' needs. These structures are independent of each other but share the same operating system (i.e., Kernel and device drivers). The example implementations are LXC (Linux Containers), Open VZ, and Solaris container [11].

We examine both unrestrained and time deadline restricted cases for the multi-objective optimization issue in modeling task scheduling. HEFT-T (Heterogeneous Earliest Finish Time- TOPSIS) algorithm is recommended that merges the HEFT method with TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution) method to solve the multi-objective problem. A three-step approach depends on the HEFT-T method is offered in the unrestrained case to choose the best possible solutions concerning the Non-dominated Sorting method. For the deadline-restrained case, an adaptive weight adjustment plan depends on HEFT-T is projected to adjust the weight rate for the time [12]. The main contributions of this paper are as follows:

- To design the multi-objective optimization problem, consider main scheduling objectives such as energy consumption, computational time, and resource utilization.
- ANFIS rule is applied to optimize the parameter for the black widow optimization algorithm.
- BWO technique has been included to address the task scheduling scheme that chooses the proper scheduling algorithm for every task.
- The performance evaluation of the proposed approach over datasets by employing different performance metrics.

The remainder of the paper is summarized in the following points. Sect. 2 discusses the related works. Section 3 discusses the problem formulation. The proposed approach is explained in Sect. 4. Section 5 discusses the experimental results and discussions. The conclusion is presented in Sect. 6.

2 Related Works

Nowadays, there are various projects dedicated to task scheduling in cloud computing to address multiple forms of multi-objective optimization problems. There are various kinds of schemes namely Genetic algorithm (GA), Cuckoo search optimization (CSO), Particle swarm optimization (PSO), Whale optimization algorithm (WOA), Chicken swarm optimization (CSO), bacterial foraging algorithm (BFO), etc. [13]. Kaur M and Kadam introduced the improvised multi-objective bacteria foraging optimization algorithm (MOBFOA) to manage the multi-objective scheduling issues using the Pareto-optimal front method. This method provides a range of trade-off solutions relevant to the various scheduling conditions for independent trades on heterogeneous resources. The fitness function can be enhanced based on the selection process to reduce the computational complexity of MOBFOA [14].

SMadni et al. [15] proposed a Multi-objective Cuckoo Search Optimization (MOCSO) approach for the resource scheduling issue. The goal of this algorithm is to minimize cloud usage costs and increase efficiency by decreasing makespan time, which helps to boost revenue or benefit for cloud providers with the highest resource consumption in the cloud computing environment of IaaS. It is appropriate for cloud computing because of its efficient resource usage via the minimum cost and time of makespan. The cuckoo search algorithm is hybridized with optimized and meta-heuristic approaches proved to be successful for the cloud computing environment. Reddy GN and Kumar SP proposed the whale optimization algorithm (WOA) for task scheduling in the cloud environment. It schedules the task that depends on fitness constraint. The fitness constraint was based on three main-constrictions: energy, resource utilization, and quality of service. And the scheduled task depends on the above three constraints so the implementation time of the task and price occupied in the execution on virtual machines is negligible. The better QoS improves the system overall performance and therefore, makes sure the improved scheduling order for the task execution [16].

Abualigah et al. [17] introduced the new hybrid antlion optimization algorithm through elite-based differential evolution for resolving multi-objective task scheduling issues in cloud computing environments. This approach was improved by employing elite-based differential evolution as the local search method to enhance the exploration capability and to keep away from providing trapped in confined optima. The proposed method is enhanced to extend time complexity. Zhang [18] proposed a multi-objective optimization approach depends on the enhanced particle swarm is to enhance the best solution accuracy, ensure the algorithm convergence capability, and cloud computing performance is enhanced. This algorithm is improved for resolving the complexity of the multi-objective optimization issue.

Srichandan et al. [19] discovered the task scheduling algorithm by utilizing a hybrid method that joins required features of the most extensively utilized biologically encouraged heuristic algorithms, the genetic algorithms (GAs), and the bacterial foraging (BF) algorithms in cloud computing. The proposed hybrid algorithm reduces the makespan, energy consumption, and also both the ecological and economic perspectives. The drawback of this approach was it takes more time for task evaluation and poor performance. Alkayalet al. [20] developed a new multi-objective PSO (MOPSO) algorithm which depends on the novel grading scheme. This approach is the new model for the resource allotment which optimizes task scheduling utilizing a multi-objective optimization as well as particle swarm

optimization. The main goal of this objective is to reduce the waiting time and increase device throughput by the scheduled tasks to the VM's.

Li et al. [21] proposed a task scheduling algorithm that depends on the Analytic network process (ANP) for the issue of multi-objective optimization in the cloud computing environment. It is utilized to resolve the issues in a task scheduling like shortage in the numerical description, restricted optimization capabilities of the conventional MOO algorithm, and Pareto best solutions selection. The enhanced gene expression programming (GEP) integrated with Non-dominated sorting genetic algorithm II (NSGA II) to enhance the worldwide optimization capability. Liu [22] proposed an algorithm called Multiopt which is the multi-objective container scheduling algorithm. The algorithm consists of five key aspects: node utilization of CPU, link among the containers and nodes, memory utilization of each node, containers clustering, consumption of time forwarding images on the system, that change the application performance in containers. The drawback of this approach was in the multiopt algorithm the containers have the fault tolerance and the rapid container migration without affecting the utilization of cloud computing services was difficult [23–33].

Zouache et al. [34] presented the chicken swarm optimization (CSO) for resolving the multi-objective optimization issues. This approach aims to balance the diversity and convergence once the best Pareto solutions are searched. This multi-objective CSO is expanded to resolve the discrete issues like multi-objective knapsack issue. Guerrero et al. presented a genetic algorithm by non-dominated sorting genetic algorithm-II (NSGA-II) to optimize the container allotment and elastic management aimed at the optimum results achieved by this algorithm for the optimization problems in cloud computing environments. This approach improves system performance, network overhead, and system failure. The main drawback of this approach was the container live relocation costs as the factor to choose solutions from the Pareto front [35].

3 Problem Definition

The main issue in the cloud computing environment is to recognize the trade-off among the user constraints and resource exploitation. Most users provide multiple tasks with different prerequisites such as computational time, memory usage, energy usage, time delay, data transfer, response time, etc., and these are considered the complexities of modeling and assessing the task scheduling in the cloud computing environment. The parameters that are crucial for the scheduling algorithms are start-up-time, execution time, scheduling cost, and energy. The other issue that is faced in scheduling is that it requires more time in the computation when dealing with an increased number of user demands in the cloud computing environment. Hence, optimization algorithms can be used to decrease the computational cost, energy, overall cost, etc.

3.1 System Model

We consider the container-based cloud data center model by the computing servers set which can accommodate containers and virtual machine (VM) occasions as per the necessities of the task and is shown in Fig. 1. Consider there are two types of users like IoT based as well as Non-IoT based who can forward the several kinds of applications or tasks (eg. request-based tasks, memory-intensive tasks, event-driven tasks) to the cloud data

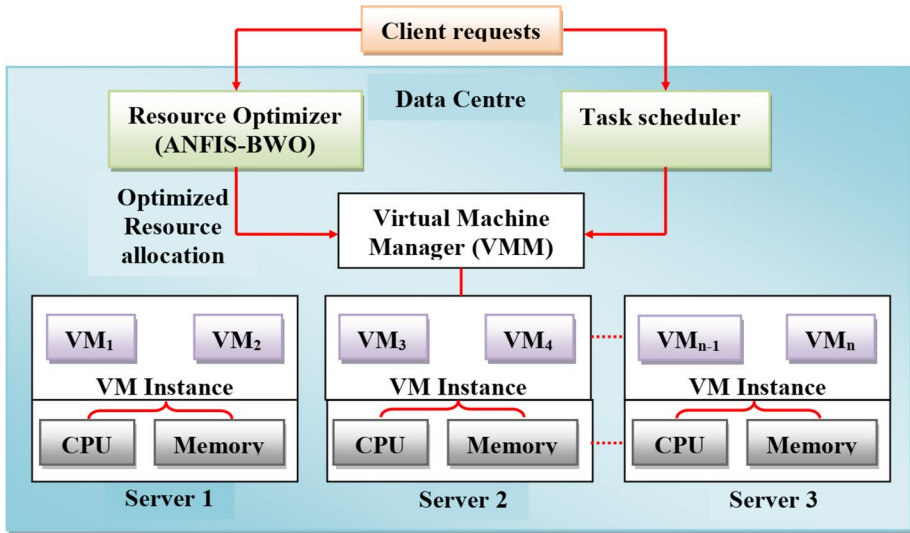


Fig. 1 Data center model of the cloud

center for processing. The action of the admission controller is to allocate the task controller either to the container schedule or virtual machine manager for additional processing [36].

The main responsibility for container-based scheduling is to create several kinds of containers depending on the necessity of the tasks. The task scheduler is dependable for grasping specific information and accepts the accessibility of the resources in the servers in every time interval. This is useful to evaluate the best server with the minimum load for organizing the chosen containers. The task schedule is responsible for auto-scaling the servers depends on the availability of the overall resource and the task arrival rate in the cloud data center. The cloud environment comprises of the service providers' numbers to offer services and users infrastructures as the request. When more number of requests enter the cloud demanding for similar resources, there exists the resource shortage and issue is considering the resource allocation to the users. Every virtual machine acquires several configurations namely memory, cost and size for tasks execution, and so on.

Hence, the accurate virtual machine allocated which acquires less cost and time for task execution is the major problem. Let us consider the cloud comprises of q_m number of physical machines that can be represented as

$$P_m = \{p_{m1}, p_{m2}, \dots, p_{mj}, \dots, p_{mq_m}\} \tag{1}$$

where $p_{m1}, p_{m2}, \dots, p_{mj}, \dots, p_{mq_m}$ represents the independent physical machines and p_{mj} denotes the cloud's physical machine. The virtual machines equivalent to the j th physical machine is represented as,

$$|v_m^j| = \{v_1^j, v_2^j, \dots, v_k^j, \dots, v_h^j\} \tag{2}$$

From the above equation v^j denotes the virtual machines respective to the j th physical machine, the total number of virtual machines is represented as h and v_h^j indicates the h virtual

machine equivalent to the j th physical machine. The cost of the resource for the k th virtual machine occurs in the j th physical machine is indicated as S_k^j . c_k^j represents the capacity of the k th virtual machine that occurred in the j th physical machine. The total task is indicated as T_i and the total task number is represented as,

$$T_i = \{t_{i1}, t_{i2}, \dots, t_{il}, \dots, t_{im}\} \quad (3)$$

From the above equation, the total tasks for all the independent tasks for the task execution are represented as T_i and the number of the total task is indicated as n , the l th task is denoted as t_{il} . Every task bears a number of sub-tasks and the l th task of subtask are indicated as,

$$t_{il} = \{t_{i1}^l, t_{i2}^l, \dots, t_{i1}^l, \dots, t_{im}^l\} \quad (4)$$

Form the above equation, $t_{i1}^l, t_{i2}^l, \dots, t_{i1}^l, \dots, t_{im}^l$ represents the l th task in the subtask and m represents the total subtasks number in the l th task. The task length is indicated as M and the task length based on the task number waiting for the execution of the task. Let the energy factor is indicated as ϵ_{kl} , which is the energy necessitated for the execution of the l th task in the k th virtual machine.

4 Proposed Methodology

The number of user requests is accomplished depends on their necessities in the cloud computing environment. But in the cloud computing environment, there are more resources so the proposer scheduling is difficult. There are various scheduling methodologies and the existing methods face various issues concerning resource cost, computational time, energy consumption, data transfer cost, resource utilization, and so on. The problems stated above were undertaken to utilize the proposed approach of the task scheduling which based on the black widow optimization algorithm. In the proposed method the computational cost is low, low energy consumption, low computational time, and efficient utilization of the resource.

4.1 Formulation of Multi-Objective Scheduling

The effective solution is established via the fitness calculation depends on the constraints such as energy consumption, makespan, cost, and energy utilization. The matrix with the minimum fitness value is selected for the task execution in their respective fitness order of the virtual machines. The matrix for the solution is indicated as w and w_{kl} is the value for the solution of l th task accomplished in the k th virtual machine. The matrix set is evaluated from the solution matrix which is evaluated via the division of the task length and machine capacity. The set matrix is indicated as S . The fitness evaluation is the main factor which calculates the effective virtual machine for executing the task without the performance is affected for the system in the method of task scheduling, the fitness function based on the some of the parameters such as, resource utilization, energy, makespan, cost, and QoS. For an efficient solution, the fitness function precedes the highest value.

Hence the main aim of this paper is to propose the MOBWO algorithm for plotting the tasks on virtual resources with minimum energy to reduce the time of the makespan and cost for improving resource utilization. This study believes the makespan time, utilization, and cost as the multi-objective functions for the best resource scheduling in the IaaS cloud computing environment. Hence the fitness values for the MOBWO algorithm can be estimated for cost, utilization, and makespan by using the below formulas.

4.1.1 Computational Cost

The computational cost of the proposed model is computed using the following equation:

$$f_x = \sum_{j=1}^n resource^j (c_j * t_j), \quad \forall j \in M, \quad j = 1, 2, 3, \dots, n \quad (5)$$

where c_j represents the resources cost j th per unit time and t_j represents the resource utilization time.

4.1.2 Makespan Model

The makespan model is formulated using the following equation:

$$f_y = \max \bigcup_{j=1}^n t_j, \quad \forall j \in M, \quad j = 1, 2, 3, \dots, n \quad (6)$$

where t_j represents the execution time for the particular task.

4.1.3 QoS

This is the main parameter in the cloud computing environment and it depends upon the scheduling virtual machine with high quality. This parameter mostly depends upon the two factors such as cost and time for the task execution in the virtual machine. The cost and time factors are low for enabling the excellent quality of service to the users. The factors $(1-c)$ and $(1-t)$ represents the scheduling cost and time is low.

$$QoS = \frac{1}{2} [(1 - c) + (1 - t)] \quad (7)$$

From the above equation, c represents the execution cost and t represents the time for the execution of the task in the virtual machine. The time for the execution of tasks in the virtual machine is based on the set and solution matrix. The execution time is specified as,

$$\begin{aligned} & \max [w_{kl} \times S_{kl}] \\ & t = \forall \\ & l \in n \end{aligned} \quad (8)$$

From the above equation, w_{kl} indicates the l th task solution executed in the k th virtual machine and S_{kl} indicates the l th task set value executed in the k th virtual machine. The cost of execution for the virtual machine is represented as,

$$c = \frac{1}{M} \sum_{k=1}^M y_{kl} \times [r_{kl}] \quad (9)$$

$$y_{kl} = w_{kl} \times S_{kl} \quad (10)$$

where the cost of execution is represented as c in the k th virtual machine, y_{kl} indicates the product factor that is established utilizing the w_{kl} and S_{kl} , and r_{kl} indicates the resource cost of the k th virtual machine. Hence the factor which assures the maximum QoS is the execution time, execution cost, and resource cost. These factors should grasp the lowest value for achieving the maximum quality value that enhances the overall system performance.

4.1.4 Energy

For the execution of the task, the energy should be low in the virtual machine for an efficient model. The energy of the virtual machine based on the l th task solution executed in the k th virtual machine and the energy necessitated for the l th task executed in the k th virtual machine.

$$e = \frac{1}{n \times q_2} \sum_{l=1}^n w_{kl} \times \varepsilon_{kl} \quad (11)$$

$$q_2 = \max(\varepsilon) \quad (12)$$

From the above equation, the virtual machines total tasks number is represented as n , q_2 denotes the factor which is the maximum of all the energy values in the virtual machine.

4.1.5 Utilization of Resources

The resource usage in the virtual machine is the other parameter that must be dramatically high to boost device performance. This parameter is based on the set and solution value.

$$ru = \frac{1}{n \times q_1} \times \sum_{l=1}^n w_{kl} \times \varepsilon_{kl}; \quad 1 \leq k \leq M; \quad 1 \leq l \leq n \quad \text{and} \quad q_1 = \max(S) \quad (13)$$

In Eq. (13), S represents the set matrix, the total number of tasks is represented as n , w_{kl} indicates the solution of l th task accomplished in the k th virtual machine and S_{kl} indicates the l th task set value executed in the k th virtual machine. The factor q_1 is evaluated that depends on the highest value for the set matrix. M represents the total virtual numbers in the cloud.

4.2 ANFIS Model

The adaptive neuro-fuzzy inference system was developed by Jang, 1993 refers to the integration of Fuzzy logic and artificial neural network to create the influential processing equipment [38]. In this article, two rules have been created for each input with a maximum value equal to 1 and a minimum value equal to 0. It is the multilayer feed-forward network in which every node achieves the specific function on the input signals.

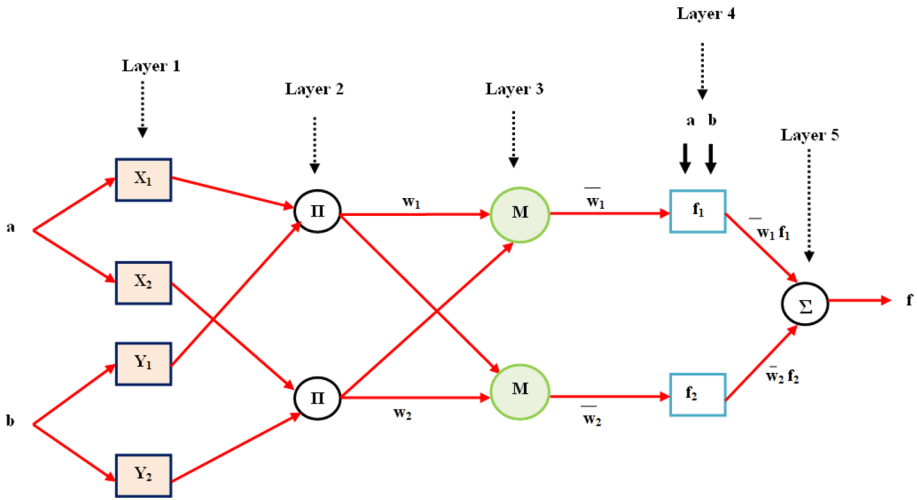


Fig. 2 ANFIS architecture

The square as well as circle node symbols utilized to characterize various learning parameters. The parameters being modified to achieve the desired input–output attributes depend on the learning rules. Fig. 2 shows the ANFIS architecture and inputs. The parameters such as computation cost, energy, memory, resource utilization, makespan are used as the input for the ANFIS model. These output parameters are tuned by using the black widow optimization algorithm to obtain optimum results. The BWO algorithm utilized in this article to assist ANFIS adjusts the parameters of the membership functions.

The fuzzy inference system is assumed which consists of 5 layers of the adaptive network with two inputs a and b , and it has only one output c . The architecture of ANFIS is shown in Fig. 2.

The node in the j th position of the l th layer is represented as $O_{l,j}$, and the functions of the node in a similar layer are of the similar function family as depicted as below:

Layer 1: Layer 1 represents the input layer and each node j in layer 1 represents the square node by the node function. $O_{1,j}$ represents the membership function of X_j and determines the degree to which the offered persuades the quantifier X_j . Generally, the bell-shaped membership function is chosen as the input of the membership function by the maximum equal to 1 and the minimum equal to 0.

$$O_{1,j} = \mu X_j(a) \quad \text{for } j = 1, 2 \tag{14}$$

$$\mu X_j(a) = \frac{1}{1 + [(a - z_j/x_j)^2]^{y_j}} \tag{15}$$

where x_j, y_j and z_j represents the parameters, y indicates the positive value and z represents the curve center.

Layer 2: Each node in this layer represents the square node, marked as Π that produces the incoming signals and forwards the output product.

$$O_{2,j} = w_j = \mu X_j(a) \times \mu Y_j(b) \quad \text{for } j = 1, 2 \tag{16}$$

Layer 3: Each node in this layer indicates the square node, marked as M. The *j*th node evaluates the ratio of the *j*th rules firing strength to the addition of all the rules for the firing strengths as per the below equation. The output of this layer shall be termed as normalized firing strengths.

$$O_{3,j} = \bar{w}_j = \frac{w_j}{w_1 + w_2} \quad \text{for } j = 1, 2 \tag{17}$$

Layer 4: Each node *j* in this layer represents the square node by the node function. The attributes in this layer represented as subsequent attributes.

$$O_{4,j} = \bar{w}_j f_j = \bar{w}_j (p_j + q_j + r_j) \tag{18}$$

where *p_j*, *q_j*, and *r_j* represents the attributes.

Layer 5: The single node in layer 5 represents the circle node, marked as Σ which evaluates the overall output is the addition of all the incoming signals as per the below equation.

$$O_{5,j} = \sum_j \bar{w}_j f_j = \frac{\sum_{j=1} w_j f_j}{\sum_{j=1} w_j} = output \tag{19}$$

4.3 Black Window Optimization Algorithm based Multi-objective Task Scheduling

In this section, the multi-objective task scheduling method represents the utilization of the proposed BWO algorithm [37]. The tasks are scheduled to the specific virtual machine depends on their execution time, energy, and execution cost. The efficiency of the scheduling algorithm depends upon the reduction in the execution time and execution cost. The BWO algorithm for the most effective virtual machine and task execution is designed to achieve the best results for the multiple user arrival requests. Similar to other traditional methods, the proposed method begins by an initialization spider population, thus every spider represents the possible solution. The initializations of spiders are in pairs and trying to produce the new generation. In this type of optimization, a female spider eats the male black spider after mating or during mating. Then the female black widow spider carries the stored sperm in her sperm theca and discharges them into egg sacs. After 11 days of being laid, spiderling comes out of the egg sacs. For several days to a week, the spiderling remains in the maternal web, in that period sibling cannibalism is determined. Consequently, the spiderlings leave the web by the wind.

- Initial Population

Through the purpose to resolve the optimization problem, the problem of variable values should form as a proper arrangement for the solution of the current problem. For PSO algorithms, this framework is called “chromosome” and “particle position,” but in black widow optimization, it is called “widow”. In BWO, the potential solution to every issue has been regarded as a black widow spider. Every black widow spider represents problem

variable values. In a K_{var} dimensional optimization issue, a widow is indicated as the array of $1 \times K_{\text{var}}$ represents the solution to the issue, and the array is described as follows:

$$\text{widow} = [u_1, u_2, \dots, u_{K_{\text{var}}}], \quad (20)$$

Every variable value is represented as $(u_1, u_2, \dots, u_{K_{\text{var}}})$ is the floating-point number. The widow fitness is evaluated by the fitness function evaluation F at the widow $(u_1, u_2, \dots, u_{K_{\text{var}}})$. Thus

$$\text{fitness} = F(\text{widow}) = F(u_1, u_2, \dots, u_{K_{\text{var}}}) \quad (21)$$

The optimization approach is initialized by creating the candidate widow matrix of size $K_{\text{var}} \times K_{\text{pop}}$ by initializing the spider's population. Then by mating, couples of parents selected arbitrarily to execute the procreating stage in that the male black widow is eaten by the female during or after mating.

- Procreate

Since the pairs are autonomous to each other, they start to mate by the purpose of reproducing the fresh generation, in similar naturally each couple mate in its web, independently from the supplementary. Almost thousands of eggs produced in every mating in the real world; however, the numbers of muscular spider babies are maintained. In this algorithm, an array is reproduced called alpha could also be produced as widow array by arbitrary numbers, including, after that, offspring are created by employing α by the subsequent equation in that u_1 and u_2 are represented as parents, whereas v_1 and v_2 represents the offspring.

$$\begin{cases} v_1 = \alpha \times u_1 + (1 - \alpha) \times u_2 \\ v_2 = \alpha \times u_2 + (1 - \alpha) \times u_1 \end{cases} \quad (22)$$

For $K_{\text{var}}/2$ times, this procedure is repeated, whereas randomly chosen numbers must not be replicated. Eventually, the mother and children are attached to the array, and the fitness is arranged, and then due to the cannibalism rating, a small number of the optimal individuals are attached to the newly produced population. These steps are provided to all pairs.

Here, the ANFIS rule is applied in this section to optimize the BWO algorithm to obtain the best results. There are four rules to be applied to optimize this approach.

If the ratio of α is less than 1, then the parameter v is low.

If the ratio of α is less than threshold t , then the parameter v is low.

If the ratio of α is equal to 1, then the parameter v is medium.

If the ratio α is greater than the threshold t , then the parameter v is high.

Where t is the threshold, the fitness value is considered as the threshold value.

- Cannibalism

In this algorithm, there are three categories of cannibalism. The initial one is sexual cannibalism, in that the after or during mating, the male widow is eaten by a female black widow. The female and male black widows are identified by their fitness values. The sibling cannibalism is another category in that the weaker siblings are eaten by the muscular spiderlings. In this algorithm, the cannibalism rating (CR) is set in accordance

with the survivor's number is evaluated. In particular conditions, the third category of cannibalism is commonly examined in that the mother spider is eaten by baby spiders. The fitness value is employed to evaluate the weak or strong spiderlings.

- Mutation

In this black widow optimization algorithm, simulated binary crossover (sbc) with a constant rate of crossover and mutation rates are utilized to produce the new chromes or individuals. An adaptive scheme is utilized to alter the mutation rate and after that, an enhanced description of mutation rate is projected. The proposed adjustive mutation operator is integrated with three operators of crossover namely, simulated binary crossover (sbc), uniform crossover (uc), and single point crossover (spc) to produce the new individuals or chromes [38]. Therefore, there are three variables of the proposed algorithm and evaluated on optimization problems.

- Fixed mutation rate

In the proposed black widow optimization algorithm, crossover and mutation are the two operators that are employed to produce newchromes in the spiderlings population. The procedure of producing individuals can be summarized in Algorithm 1. For each individual, once the implementation of the crossover operator is finished, the mutation operator starts, this is shown in Algorithm 2. In the 2nd algorithm, the mutation in the variable which indicates whether each chromes implements the mutation operator or not; R_1 and R_2 are the two arbitrarily generated numbers among 0 and 1.0.

Algorithm 1: Updating the process of the population with a fixed mutation rate

Input: Population p , mutation rate m_r , crossover rate c_r , distributive index η

Output: Population of the spiderlings p_s

For $j=1$ to p_{max} do // p_{max} is the population size

 For $j = 1$ to n_p do // n_p is the population size

 Implement the operators of the crossover (sbc, uc, spc);

 Mutation (p , m_r);

 All the updated individuals to produce spiderlings population p_s

 end for

end for

Algorithm 2: Mutation operator

Input: parent P , mutation rate m_r , distributive index η

Output: child c

$c = P$; $mutated = false$;

for $j=1$ to L do // L is the length of the c

 if $s_j \leq m_r$ then

$mutated = true$;

$b = c_j$;

$\delta_l = (b - lb) / (ub - lb)$

$\delta_r = (lb - b) / (ub - lb)$

$M = 0.1 / (\eta + 1.0)$

 if $(s_2 \leq 0.5)$ then

$a = 1.0 - \delta_l$;

$y = 2.0 * s_2 + (1.0 - 2.0 * s_2) * a^{(\eta + 1.0)}$;

$\delta_r = y^M - 1.0$;

 else

$a = 1.0 - \delta_2$;

$y = 2.0 * (1 - s_2) + (s_2 - 0.5) * a^{(\eta + 1.0)}$;

$\delta_r = 1.0 - y^M$;

 endif

$b = b + \delta_r * (ub - lb)$;

$b = \min(ub, \max(lb, b))$;

$c_j = b$;

 endif

 end for

- Adaptive mutation rate

The proposed BWO algorithm by the permanent mutation rate does not calculate the ultimate solution for the optimization issue. The adaptive scheme utilized for this paper can be utilized to undertake this problem by altering the mutation rate. The linear function is presented to alter the mutation rate for easefulness. Hence the mutation rate m_r shall be updated as follows:

$$m_r = \frac{p_s}{L} \quad (23)$$

where p_s can be evaluated as follows:

$$p_s = P + (t - 1) \times \frac{1 - P}{t_M - 1} \tag{24}$$

Form the above Eqn. t and t_M are the present and maximum generation accordingly, P denotes the fixed real number. In this article, P shall be provided as follows.

$$P_0 = \frac{L}{50} \tag{25}$$

Here L represents the problem of interest dimension.

When t is equal to 1, the starting mutation rate P_0 is 1/50 (0.02). The detailed study of the adaptive mutation rate is discussed.

Subsequent to integrating the adaptive mutation scheme into the black widow optimization algorithm, the process of producing new chromes can be updated as depicted in Algorithm 3. There are three well-identified types of crossover operators: sbc, uc, and spc. These adaptive mutation operators are combined with the three crossover operators, three new chrome updating schemes are then projected and termed as sbcam, ucam, and spcam correspondingly.

Algorithm 3: Updating process of population with adaptive mutation rate

Input: Population p , initial mutation rate P_0 , crossover rate c_r , distributive index η

Output: Population of the spiderlings p_s

For $j=1$ to p_{max} do // p_{max} is the population size

Evaluate mutation rate m_r according to Eqn. ()

For $j = 1$ to n_p do // n_p is the population size

Implement the operators of the crossover (sbc, uc, spc);

Mutation (p, m_r);

All the updated individuals to produce spiderlings population p_s

end for

end for

- Convergence

Identical to evolutionary algorithms, there are three stopping criteria can be regraded: (1) predefined iteration numbers. (2) performance of no alter in the fitness value of the optimal window for some iterations. (3) Achieving the specified accuracy level. The pseudo-code for the proposed task scheduling approach is illustrated in Algorithm 4.

Algorithm 4: Pseudocode for the task scheduling using ANFIS-BWO

```

Input: client requests, model for the multi-objective scheduling
Output: computational time, executional cost, energy consumption, optimal scheme
Initialize Population  $p$ , mutation rate  $m_r$ , crossover rate  $c_r$ , distributive index  $\eta$ , number of iterations
While  $t <$  maximum number of iterations
  for each black widow spider
    for each task
      compute the fitness value by eqn. (21)
      procreate the spiderlings by window array  $\alpha$  and offsprings  $v$ 
      for  $K_{var} / 2$  times repeat the process
        mother and children are linked to the array
        ANFIS rule is utilized to achieve the best results
      If  $\alpha < 1$ , then  $v$  is low
      If  $\alpha >$  fitness value, then  $v$  is low
      If  $\alpha = 1$ , then  $v$  is medium
      If  $\alpha >$  fitness value, then  $v$  is high
    end if;
  end if;
  end if;
  end if;
  end for;
  perform the cannibalism operation;
  update the individuals to produce spiderlings by Algorithm 1
  mutation operation is computed by Algorithm 2
  mutation rate is updated by Eqn. (23)
  update new chromosomes by Algorithm 3
end for
end for
record the optimal solution of the current iteration
 $t = t + 1$ ;
If the current iteration result is not as same as the last
  then  $k = 0$ ;
  else  $k = k + 1$ ;
end if;
end while;
Find the best scheme for the task scheduling process, bind the virtual machine to find computational time, execution cost and energy consumption.

```

5 Results and Discussions

The proposed ANFIS-BWO for task scheduling is evaluated utilizing several experiments. The implementation is executed in the PC by Intel Core i-3 processor 4 GB RAM and Windows 10 operating system. JAVA supports the entire deployment using cloud hardware. The performance evaluation of the ANFIS-BWO algorithm was evaluated over various

synthetic databases of different sizes. The four significant datasets linked with the task size are depicted in Table 1. In this experiment, the task sizes are created arbitrarily at the runtime and its size is indicated by the millions of instructions (MI).

5.1 Performance Comparison

In this part, the simulation results of the proposed algorithm for the performance metrics such as computation time, cost, resource utilization, makespan, energy consumption, memory utilization is compared with various existing algorithms like EECS [41], LB-RC [40], and ANFIS [42].

- Computational time:** The computational time for the task is described as the total time required for the task execution. It is the time required to complete its finishing the task inside the particular device. The proposed ANFIS-BWO algorithm estimates the appropriate device for every task and also finds the optimal server. The time depends on the time the allocated resources complete the particular task, the time required to forward the task from the device to the cloud data center, as well as the time needed to arrange the executing device. Some of the approaches require more time for longer tasks. This will increase the completion time for the task execution. But for the proposed method, the computational time for task execution is low, whereas high resource capacity tasks are allocated to the most efficient VM requests, as opposed to less efficient VM instances. The proposed approach requires low computational time when compared with other algorithms such as EECS, LB-RC, and ANFIS because of using a larger amount of resources.

The average computation time for the proposed ANFIS-BWO approach takes less time for the task execution by 11% for LB-RC, EECS by 15% and for ANFIS by 25%. Depends on the experiments, the proposed approach achieves low computation time compared to other approaches for various datasets and is shown in Fig 3.

- Makespan:** The makespan of the task set is described as the total completion time for the set of the task reached at the particular time illustration. The experimental evaluations of the four methods over the four datasets are illustrated in Fig. 4. The lesser computational time for the tasks will complete their execution faster, which generates the minimum makespan. The makespan value raises along with the number of tasks increases in the sets of tasks. This will decrease the capability to hold the server tasks and makespan is increased. The proposed approach finds the best solution and organizes the task to the best-fit virtual machine examples that will decrease the execution time and also the makespan.

Table 1 Synthetic datasets sizes

Tasks types	Task numbers	Task size (MI)
Minimum	900–1000	100,000–200,000
Mean	700–800	80,000–100,000
Maximum	500–600	50,000–70,000
SD	100–300	30,000–50,000

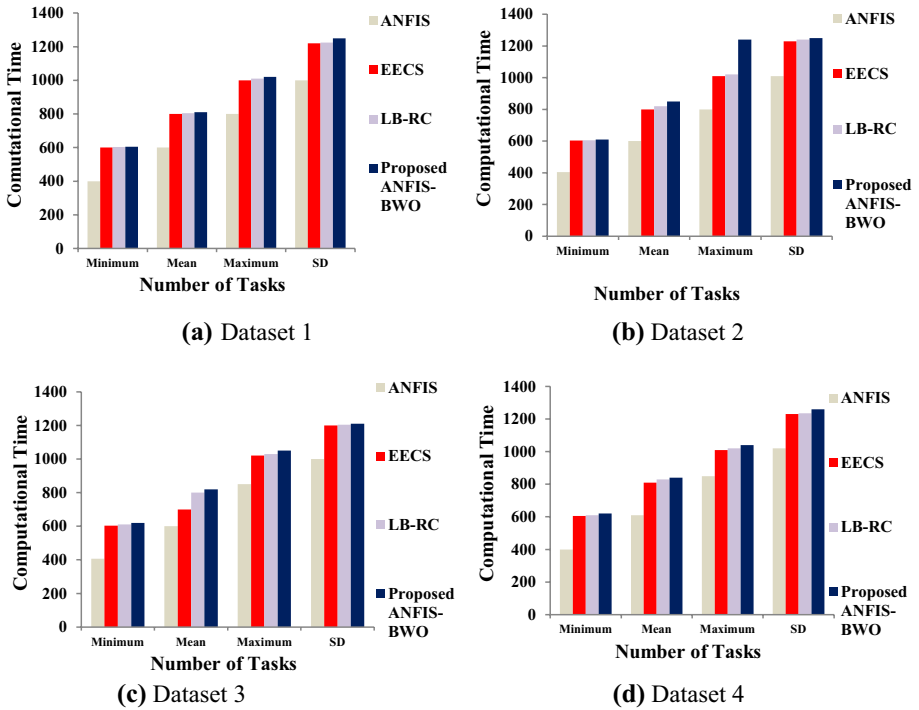


Fig. 3 Computational time

- The makespan for the proposed ANFIS-BWO approach is better than the LB-RC by 16%, EECS algorithm by 21%, and ANFIS by 34%. Depends on the experiments, the proposed approach minimizes the makespan of the tasks compared to other approaches for various datasets.
- **Energy Consumption:** The energy consumption of the task is based on the energy used by the forwarding channel and the resources when the task is processed in the computing server. In this, the device must utilize the minimum amount of resources that will decrease the overall energy consumption for task completion. The proposed ANFIS-BWO approach discovers the appropriate device for every task and assigns it to the optimal server. The other approach such as EECS and LB-RC employs the tasks for the appropriate virtual machines that will consume high energy for the task execution. For the different approaches, metrics such as maximum value, mean, minimum value, and standard deviation (SD) are denoted and evaluated depending on the effects of several databases' energy consumption. The other approaches such as EECS, LB-RC, and ANFIS performances are worse than our proposed approach.

The average energy consumption for the proposed ANFIS-BWO approach is better than the LB-RC by 22%, EECS algorithm by 28%, and ANFIS by 31%. Depends on the experiments, the proposed approach achieves less energy consumption when compared to other approaches for various datasets is shown in Fig 5.

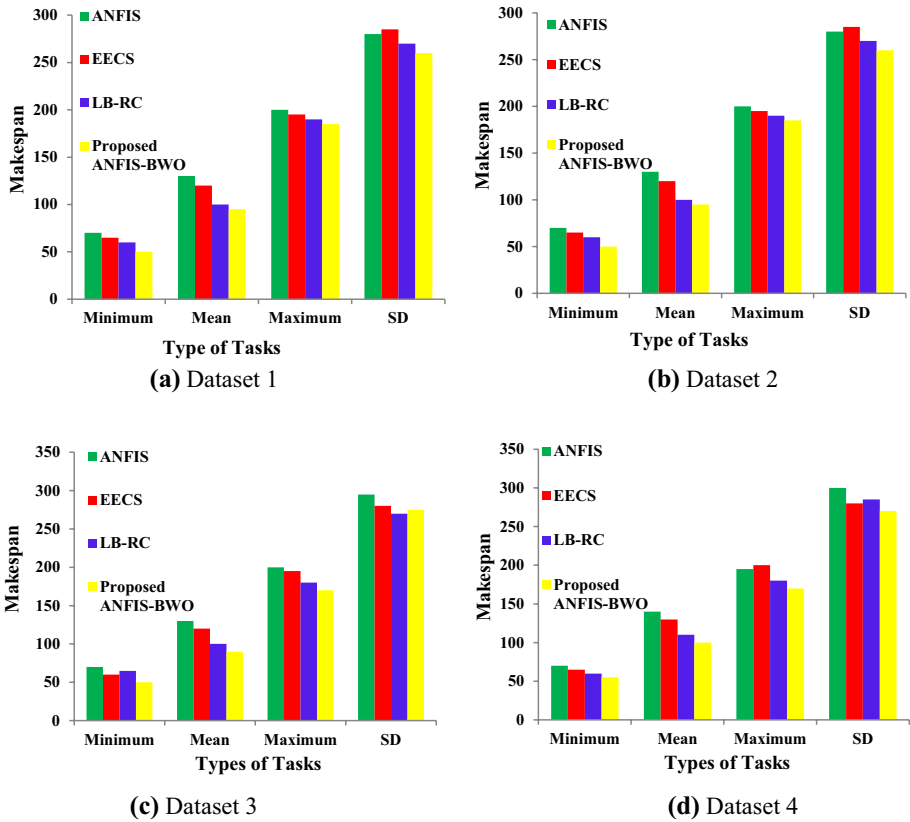


Fig. 4 Makespan

- Computational cost:** Computational cost is the total cost for the task execution and this parameter will affect the overall performance of the system. Computational cost indicates the total execution task for a particular task on specific virtual machines. The computational cost depends upon the data transfer cost, storage, and task length. When there are more tasks, the global best solution will be achieved, since a large number of tasks and limited resources tend to increase costs, waiting time, and makespan. The proposed ANFIS-BWO algorithm is cost-effective with respect to the other algorithms evaluated against the various kinds of tasks. The computational cost for the proposed ANFIS-BWO approach is better than the LB-RC algorithm by 18%, EECS algorithm by 22%, and ANFIS by 33%. The experiments describe that the proposed approach reduces the computation cost of the tasks. The evaluation for the computation cost for various approaches in different datasets is described in Fig. 6.
- Resource utilization:** The resource utilization of the server indicates the total time required for the task execution and also to reschedule the resources for the execution of the tasks in the cloud data center. The rescheduling of these resources can reduce the resource usage and number of servers used to perform the requested task. The virtual machines have their own operating system in this process and need extra memory to store an operating system memory file for an additional CPU cycle. This

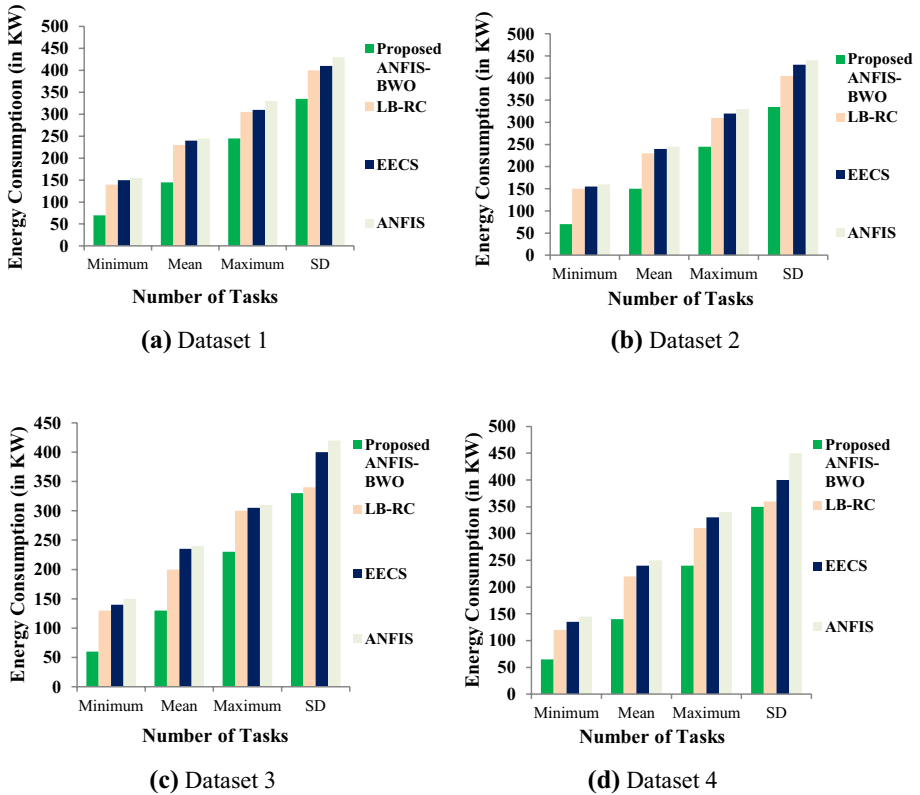


Fig. 5 Energy consumption

will decrease the parallelism among the tasks in the server and decreases resource usage in terms of memory and CPU. But the device completes more tasks and parallelism is increased between the tasks. This scheme again enhances the memory and CPU utilization for all the servers. The proposed approach establishes the proper device instead of the virtual machines for every task that fits for the best server.

The comparative examination among the proposed ANFIS-BWO approach and the other state-of-art-approaches in terms of CPU and memory usage for several databases are shown in Fig. 7. The higher resource utilization will complete the number of tasks in parallel on the CDC. The memory utilization of the ANFIS-BWO approach is better than the LB-RC approach by 19%, EECS by 25%, and ANFIS by 35%. The enhancements in the proposed approach for memory utilization over other approaches in different datasets are illustrated in Fig. 7.

The CPU utilization of the ANFIS-BWO approach is better than the LB-RC approach by 14%, EECS by 19%, and ANFIS by 26%. The enhancements in the proposed approach for the CPU utilization over other approaches in various datasets are illustrated in Fig. 8.

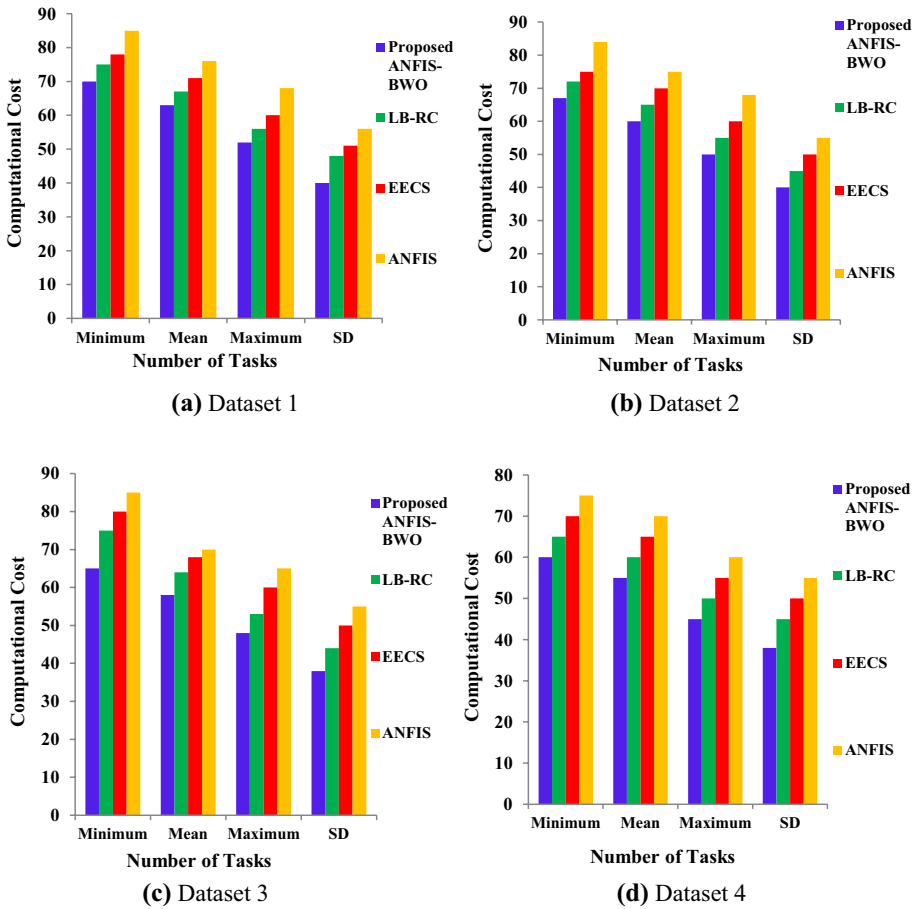


Fig. 6 Computational cost

6 Conclusions

The proposed approach uses an ANFIS-BWO method to establish the right VM with lesser delay. In this work, we have developed the fuzzy-based black widow optimization algorithm to obtain the optimal parameters in the cloud server. The main contribution of this method is to create the appropriate virtual machines for each task, based on multi-objective such as energy usage, computational time, computational cost, makespan, and resource utilization. This method decreases energy consumption, makespan, computational time, and offers efficient resource utilization. The algorithm establishes a suitable server based on the ruling scheme for the virtual machine for better resource utilization such as CPU and memory. The large-scale simulation experiments are built over four

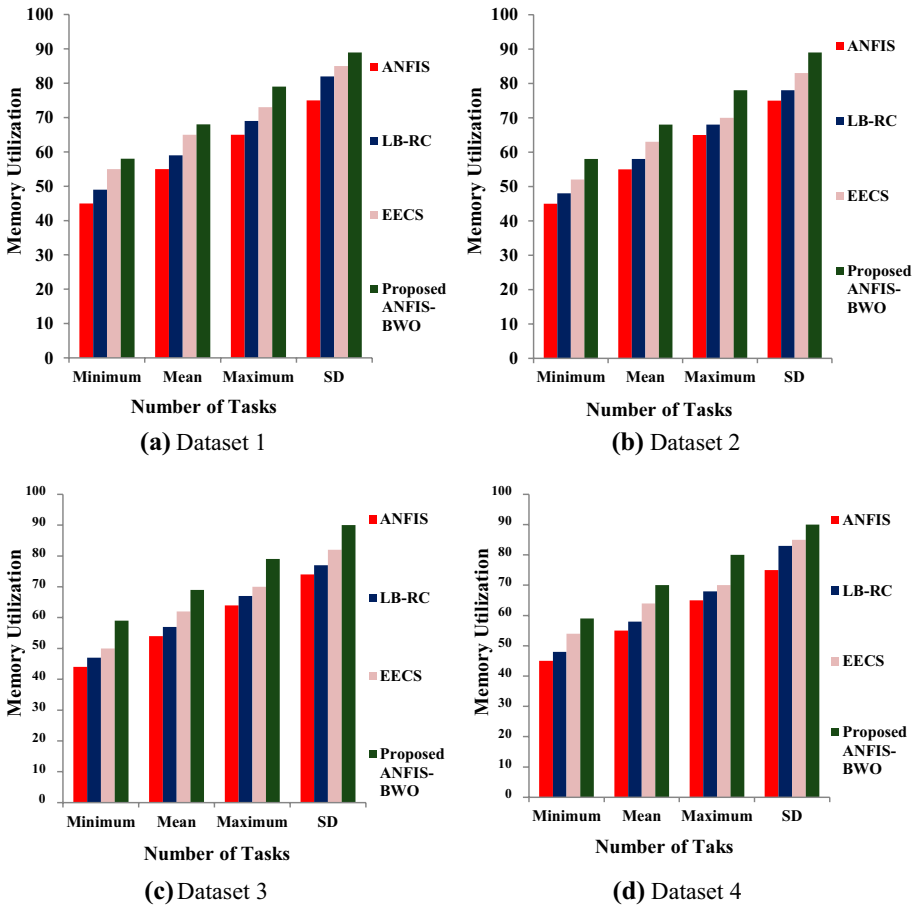


Fig. 7 Memory utilization

different synthetic datasets which show that the proposed approach is better than the current state-of-the-art approaches.

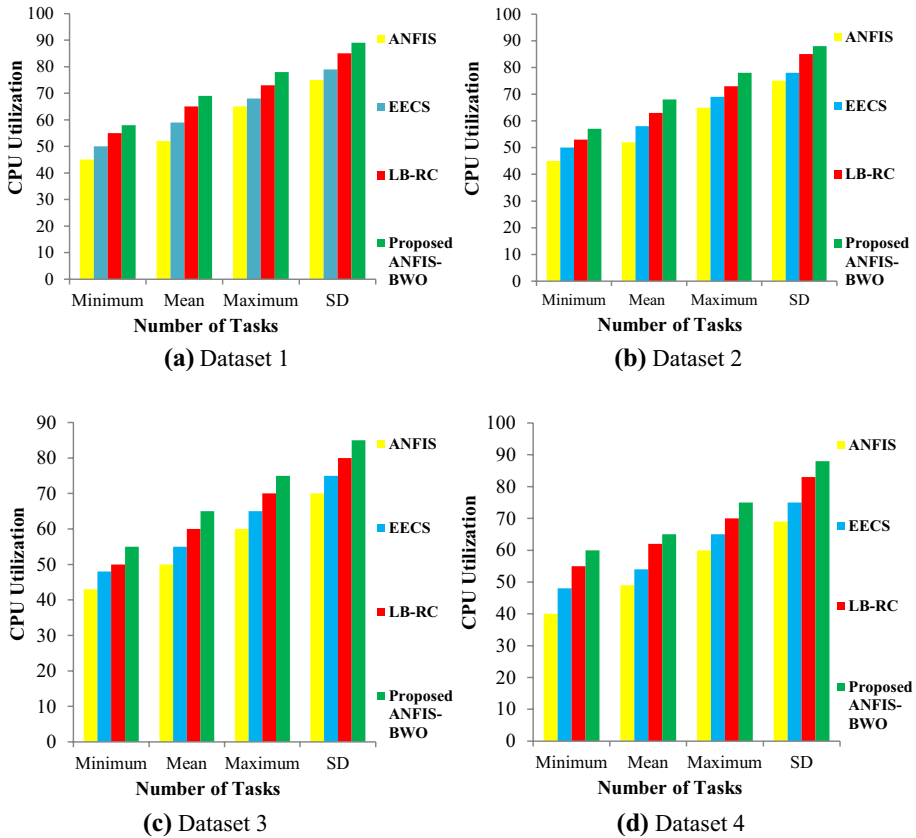


Fig. 8 CPU utilization

References

1. Stergiou, C., Psanniss, K. E., Kim, B. G., & Gupta, B. (2018). Secure integration of IoT and cloud computing. *Future Generation Computer Systems.*, 1(78), 964–975.
2. Buyya, R., Vecchiola, C., & Selvi, S. T. (2013). *Mastering cloud computing: Foundations and applications programming*. Newnes.
3. Noor, S., Koehler, B., Steenson, A., Caballero, J., Ellenberger, D., & Heilman, L. (2019). IoTDoc: A docker-container based architecture of IoT-enabled cloud system. In *3rd IEEE/ACIS international conference on big data, cloud computing, and data science engineering 2019* May 29 (pp. 51–68). Springer.
4. Kim, N. Y., Ryu, J. H., Kwon, B. W., Pan, Y., & Park, J. H. (2018). CF-CloudOrch: Container fog node-based cloud orchestration for IoT networks. *The Journal of Supercomputing.*, 74(12), 7024–7045.
5. Luo, J., Yin, L., Hu, J., Wang, C., Liu, X., Fan, X., & Luo, H. (2019). Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT. *Future Generation Computer Systems.*, 1(97), 50–60.
6. Pandi, V., Perumal, P., Balusamy, B., & Karuppiyah, M. (2019). A novel performance enhancing task scheduling algorithm for cloud-based E-health environment. *International Journal of E-Health and Medical Communications (IJEHMC).*, 10(2), 102–117.
7. Tang, Z., Qi, L., Cheng, Z., Li, K., Khan, S. U., & Li, K. (2016). An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. *Journal of Grid Computing.*, 14(1), 55–74.

8. Choi, S., Myung, R., Choi, H., Chung, K., Gil, J., & Yu, H. (2016). December. Gpsf: general-purpose scheduling framework for container based on cloud environment. In *2016 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)* (pp. 769–772). IEEE.
9. Patra, M. K., Patel, D., Sahoo, B., & Turuk, A. K. (2020). Game theoretic task allocation to reduce energy consumption in containerized cloud. In *2020 10th international conference on cloud computing, data science and engineering (confluence)* (pp. 427–432). IEEE.
10. Wu, S., Niu, C., Rao, J., Jin, H., & Dai, X. (2017). Container-based cloud platform for mobile computation offloading. In *2017 IEEE international parallel and distributed processing symposium (IPDPS)* (pp. 123–132). IEEE.
11. Canosa, R., Tchernykh, A., Cortes-Mendoza, J. M., Rivera-Rodriguez, R., Rizk, J. L. Avetisyan, A., & Concepcion Morales, E. R. (2018). Energy consumption and quality of service optimization in containerized cloud computing. In *2018 IvannikovIspras open conference (ISPRAS)*. <https://doi.org/10.1109/ispras.2018.00014>.
12. Liu, L., Fan, Q., & Buyya, R. (2018). A deadline-constrained multi-objective task scheduling algorithm in mobile cloud environments. *IEEE Access*, *18*(6), 52982–52996.
13. Hu, H., He, J., He, X., Yang, W., Nie, J., & Ran, B. (2019). Emergency material scheduling optimization model and algorithms: a review. *Journal of Traffic and Transportation Engineering (English edition)*, *6*, 441–454.
14. Kaur, M., & Kadam, S. (2018). A novel multi-objective bacteria foraging optimization algorithm (MOBFOA) for multi-objective scheduling. *Applied Soft Computing*, *1*(66), 183–195.
15. Madni, S. H., Latiff, M. S., & Ali, J. (2019). Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds. *Arabian Journal for Science and Engineering*, *44*(4), 3585–3602.
16. Reddy, G. N., Kumar, S. P. (2017). Multi objective task scheduling algorithm for cloud computing using whale optimization technique. In *International conference on next generation computing technologies* (pp. 286–297). Springer.
17. Abualigah, L., & Diabat, A. (2020). A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Cluster Computing*, *24*, 1–19.
18. Zhang, M., & Li, G. (2018). Multi-objective optimization algorithm based on improved particle swarm in cloud computing environment. *Discrete and Continuous Dynamical Systems-S*, *12*(4 & 5), 1413.
19. Srichandan, S., Kumar, T. A., & Bibhudatta, S. (2018). Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. *Future Computing and Informatics Journal*, *3*(2), 210–230.
20. Alkayal, E. S., Jennings, N. R., & Abulkhair, M. F. (2016). Efficient task scheduling multi-objective particle swarm optimization in cloud computing. In *2016 IEEE 41st conference on local computer networks workshops (LCN workshops)* (pp. 17–24). IEEE.
21. Li, K., & Wang, J. (2017). Multi-objective optimization for cloud task scheduling based on the ANP model. *Chinese Journal of Electronics*, *26*(5), 889–898.
22. Liu, Bo., Li, P., Lin, W., Shu, Na., Li, Y., & Chang, V. (2018). A new container scheduling algorithm based on multi-objective optimization. *Soft Computing*, *22*(23), 7741–7752.
23. Hassan, B. A. (2021). CSCF: A chaotic sine cosine firefly algorithm for practical application problems. *Neural Computing and Applications*, *33*(12), 7011–7030. <https://doi.org/10.1007/s00521-020-05474-6>.
24. Sundararaj, V. (2016). An efficient threshold prediction scheme for wavelet based ECG signal noise reduction using variable step size firefly algorithm. *International Journal of Intelligent Engineering and Systems*, *9*(3), 117–126.
25. Sundararaj, V., & Selvi, M. (2021). Opposition grasshopper optimizer based multimedia data distribution using user evaluation strategy. *Multimedia Tools and Applications*. <https://doi.org/10.1007/s11042-021-11123-4>.
26. Alam, M.G., & Baulkani, S. (2019). Local and global characteristics-based kernel hybridization to increase optimal support vector machine performance for stock market prediction. *Knowledge and Information Systems*, *60*(2), 971–1000.
27. Nirmal Kumar, S. J., Ravimaran, S., & Alam, M. M. (2020). An effective non-commutative encryption approach with optimized genetic algorithm for ensuring data protection in cloud computing. *Computer Modeling in Engineering & Sciences*, *125*(2), 671–697.
28. Jose, J., Gautam, N., Tiwari, M., Tiwari, T., Suresh, A., Sundararaj, V., & Rejeesh, M. R. (2021). An image quality enhancement scheme employing adolescent identity search algorithm in the NSST domain for multimodal medical image fusion. *Biomedical Signal Processing and Control*, *66*, 102480.

29. Nisha, S., & Madheswari, A. N. (2016). Secured authentication for internet voting in corporate companies to prevent phishing attacks. *International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)*, 22(1), 45–49.
30. Albert, P., & Nanjappan, M. (2020). An efficient kernel FCM and artificial fish swarm optimization-based optimal resource allocation in cloud. *Journal of Circuits, Systems and Computers*, 29(16), 2050253.
31. Nirmal Kumar, S. J., Ravimaran, S., & Alam, M. M. (2020). An effective non-commutative encryption approach with optimized genetic algorithm for ensuring data protection in cloud computing. *Computer Modeling in Engineering & Sciences*, 125(2), 671–697.
32. Sundararaj, V. (2019). Optimal task assignment in mobile cloud computing by queue based ant-bee algorithm. *Wireless Personal Communications*, 104(1), 173–197.
33. Nanjappan, M., & Albert, P. (2019). Hybrid-based novel approach for resource scheduling using MCFCM and PSO in cloud computing environment. *Concurrency and Computation: Practice and Experience*, e5517.
34. Zouache, D., Arby, Y. O., Nouioua, F., & Abdelaziz, F. B. (2019). Multi-objective chicken swarm optimization: A novel algorithm for solving multi-objective optimization problems. *Computers and Industrial Engineering*, 129, 377–391.
35. Guerrero, C., Lera, I., & Juiz, C. (2018). Genetic algorithm for multi-objective optimization of container allocation in cloud architecture. *Journal of Grid Computing*, 16(1), 113–135.
36. Reddy, G. N., & Kumar, S. P. (2017). Multi objective task scheduling algorithm for cloud computing using whale optimization technique. In *International conference on next generation computing technologies* (pp. 286–297). Springer
37. Hayyolalam, V., & Kazem, A. A. (2020). Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems. *Engineering Applications of Artificial Intelligence*, 87, 103249.
38. Yi, J. H., Deb, S., Dong, J., Alavi, A. H., & Wang, G. G. (2018). An improved NSGA-III algorithm with adaptive mutation operator for Big Data optimization problems. *Future Generation Computer Systems*, 1(88), 571–585.
39. Jang, J. S. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3), 665–685.
40. Adhikari, M., Nandy, S., & Amgoth, T. (2019). Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud. *Journal of Network and Computer Applications*, 15(128), 64–77.
41. Adhikari, M., & Srirama, S. N. (2019). Multi-objective accelerated particle swarm optimization with a container-based scheduling for Internet-of-Things in cloud environment. *Journal of Network and Computer Applications*, 1(137), 35–61.
42. López-Santana, E., Méndez-Giraldo, G., Figueroa-García, J. C. (2019) Scheduling in queueing systems and networks using ANFIS. In *Uncertainty management with fuzzy and rough sets* (pp. 349–372). Springer.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Dr. Manikandan Nanjappan obtained his Bachelors of Engineering degree in Computer Science Engineering from Anna University 2005. Then he obtained his Master's of Engineering degree in in Computer Science Engineering from Anna University 2011. He completed his PhD in Sathyabama Institute of Science and Technology 2020, he is an assistant professor of Computer Science and engineering, St joseph's college of engineering. His specializations include cloud computing, Operating systems, networking.



Dr. Gobalakrishnan Natesan pursued his Bachelor's degree in Information Technology at Anna University, Tamilnadu, India in 2005. He then obtained his Master's degree in Software Engineering from Bharathidasan University, Tamilnadu, India in 2008. He received his Ph.D from Sathyabama Institute of Science and Technology, Chennai, India and working as Associate Professor in the Department of Information Technology, Sri Venkateswara College of Engineering, Chennai, Tamilnadu, India. His current research interests are Cloud computing, IoT and Big Data. He is a life time member of ISTE.



Dr. Pradeep Krishnadoss pursued his Bachelor's degree in Computer Science and Engineering at Madras University, Tamilnadu, India in 2004. He then obtained his Master's degree in Computer Science and Engineering from Anna University, Tamilnadu, India in 2011. He completed his Ph.D in Sathyabama Institute of Science and Technology, Chennai, India and working an Assistant Professor in the School of Computer Science and Engineering, VIT University, Chennai, Tamilnadu, India. His current research interests are Cloud computing, Virtualization and Big Data.