Cluster Computing (2024) 27:7621–7636
https://doi.org/10.1007/s10586-024-04378-7

Task scheduling using fuzzy logic with best-fit-decreasing for cloud computing environment

Nitin Thapliyal[1] · Priti Dimri[2]

Received: 5 December 2023 / Revised: 21 February 2024 / Accepted: 22 February 2024 / Published online: 28 March 2024

Abstract
An efficient task scheduling is mandatory in cloud computing for providing virtual resources used to carry out the tasks. An effective allocation of VM with the presence of diverse resource requirements, inaccurate information and uncertainties existing in the system is difficult. In this research, an effective task scheduling is done by using the fuzzy logic (FL) with best-fit-decreasing (BFD) in a cloud computing environment. The developed FL–BFD is optimized using resource usage, power, cost and time. Accordingly, the FL–BFD reallocates virtual machine (VM) in the cloud, based on the user demands. Therefore, the adaptability of FL is leveraged to handle uncertainties and imprecise information, which is helpful for an appropriate allocation of VM using BFD according to user requirements. The developed FL–BFD is analyzed using makespan, execution time, degree of imbalance, energy consumption and service level agreements (SLA) violations. The existing approaches named minimum completion time (MCT), particle swarm optimization (PSO), improved wild horse optimization with levy flight algorithm for task scheduling in cloud computing (IWHOLF-TSC), inverted ant colony optimisation (IACO), fuzzy system and modified particle swarm optimization (FMPSO), and task-scheduling using whale optimization (TSWO) are used for comparison. The makespan of FL–BFD with 1000 tasks is 9.2 ms, which is higher when compared to the IWHOLF-TSC and MCT-PSO.

Keywords Best-fit-decreasing · Cloud computing · Fuzzy logic · Makespan · Resource usage · Service level agreements · Virtual machine · Task scheduling

1 Introduction

In the recent times, there has been an extensive growth in technological advancements which demand high-performing computing systems for accomplishing the tasks. Therefore, cloud computing is developed as a necessary solution to simplify and provide efficient resources for performing difficult tasks [1]. Cloud computing is a revolution in information technology (IT) and other fields due to its efficient and powerful structure. Cloud is considered a major key for big scale data and complex computing operations [2, 3]. Different cloud features such as quality of service, on demand, virtualization, elasticity, usage-based billing and self-service have led to an increase in the concentration of studies about cloud in research communities and industries. Moreover, cloud computing provides better outcomes in managing data and infrastructure [4]. It has three types of delivery models namely, software as a service (SaaS), infrastructure as a service (IaaS) and platform as a service (PaaS), as well as four deployment models namely, private, public, hybrid, and community cloud [5]. The SaaS denotes the service for users via system interfaces, PaaS denotes the cloud's operating system, and IaaS has hardware amenities such as network and storage. The PaaS operates the data collection between IaaS and SaaS through network facilities [6]. The cloud architecture is a network of parallel and distributed

✉ Nitin Thapliyal
thapliyal.nitin@gmail.com

Priti Dimri
FACT0090054@uktech.net.in

1 Department of Computer Science and Engineering, Veer Madho Singh Bhandari Uttarakhand Technical University, Dehradun, Uttarakhand, India

2 Department of Computer Science and Applications, G.B. Pant Engineering College, Ghurdauri, Pauri, Uttarakhand 246194, India

systems, created using a group of virtual machines (VMs) that provide computing resources with respect to the service-level agreement, which is an agreement between clients and the service providers [7].

The cloud computing is an IT deployment system utilized to share resources comprising of analytics, memory, intelligence, information, software, hardware, servers, networking, video, audio, desktop accessibility, storage space, printers, web services, emails, applications, etc., [8]. Cloud service providers (CSPs) provide and accomplish the services according to the requirements of the customers. The customers need VMs for installing the applications at the datacenter by using CSPs [9, 10]. High flexibility levels are obtained by the end users through the dynamic nature of cloud with on-demand resources, but they also make resource management highly complex. Task scheduling is an essential requirement for accomplishing resource management and performance optimization in cloud computing structures [11–13]. In cloud computing, task scheduling is the process of task assignment to the users according to the existing resources for enhancing resource efficiency, improving load balancing and optimizing operation time. Task scheduling is mainly based on the dependencies between tasks [14]. The tasks are allocated to the resources with stronger a calculating power to decrease the task completion time which creates the issue of deficiency in load balancing. Hence, the balance between load balancing and task completion time is required to be considered for cloud architecture [15].

An appropriate task scheduling performs an important role in optimizing the resource utilization, improving system efficiency and reducing the task completion time in cloud computing environments. But, the challenge here is an effective allocation of VM according to its diverse resource requirements, existence of uncertainties and inaccurate information in the system. This issue is considered as a motivation for this research work. The main objective of this research is to enhance the efficiency of task scheduling in cloud computing by combining FL with the BFD algorithm. The goal is to leverage the adaptability of FL for handling uncertainties and imprecise information that concurrently support appropriate allocation of VM using BFD as per the user requirements.

This research makes the below contributions:

- The integration of FL–BFD enables an effective VM reallocation based on the needs of the users. Here, the FL–BFD is optimized by various factors such as resource usage, power, cost and time. Therefore, the integrated approach of FL–BFD optimizes resource usage according to the FL's adaptability and efficiency of BFD. FL takes precise decision by considering the dynamic behavior of cloud environments, whereas the FL–BFD confirms that the tasks are assigned in a way that reduces the resource usage.
- The selection of VM according to the task's resource requirements helps to reduce the execution time which contributes to the overall system efficiency and its responsiveness. Thus, the system efficiency is enhanced by adapting uncertainties, optimizing the resource usage and reducing the execution time using FL–BFD.

Rest of the research is structured as follows: Sect. 2 provides the related works of task scheduling in the cloud computing environment. The detailed explanation about FL–BFD-based task scheduling via VM reallocation is given in Sect. 3, whereas the outcomes are presented in Sect. 4. Finally, a conclusion is drawn in Sect. 5.

2 Related work

The related works of task scheduling in cloud computing environment along with their advantages and limitations are provided in this section.

Ghafari and Mansouri [16] presented the multi-objective task scheduling namely DCOHHO task scheduling (DCOHHOTS) according to the modified Harris hawks optimizer (HHO). The optimal configuration was chosen from opposition-based learning, chaotic map and population ratio. This optimal configuration was used for initializing the location of hawk for HHO. Further, the developed DCOHHOTS was used to optimize the resource usage for minimizing energy, cost and makespan. Before submitting to the scheduler, task prioritization was achieved by using the hierarchical procedure in the DCOHHOTS. However, load balancing over the cloud was further needed to be enhanced for an effective cloud computing architecture.

Malathi and Priyadarsini [17] developed the hybrid lion-based genetic method for choosing the VM. The task and VM selection probability were utilized for analyzing the Physical Machines (PMs) that were underloaded and overloaded in cloud. The multi-objective task scheduling was used to accomplish the selection probability. Further, the lion optimizer redefined the local search values of load and VM capacity. Next, the genetic operators were deployed for deriving the global optimal solution. Thus, the tasks were scheduled and executed by considering the load, task execution cost and capacity of the hybrid lion-based genetic method. An appropriate selection of VMs was required for further optimizing the resource utilization.

Lipsa et al. [18] presented the M/M/n queuing approach to perform task scheduling in cloud computing architecture. The priority of individual tasks was allocated by designing the waiting time matrix using the priority assignment method. Next, the task with higher priority was extracted by implementing a unique idea according to the Fibonacci heap using the waiting queue. This work developed a parallel algorithm to perform task scheduling, where task priority allocation and heap creation were ensured in a parallel manner with the consideration of preemptive and non-preemptive scheduling methods. However, even when the tasks were allocated with the priority of resources, the execution time was affected due to heap constructions.

Saroit and Tarek [19] developed the Hungarian method to solve the issue of load balancing over the cloud. The Hungarian method was utilized for solving the weighted matching issues, discovering the perfect matching among the resources and challenging alternatives, therefore less cost and higher profit were achieved during the load balancing process. The developed Hungarian method did not consider the priority and dependencies of the tasks in cloud environment.

Emami [20] presented enhanced sunflower optimization (ESFO) to enhance the task scheduling performances. The developed ESFO attained the optimum scheduling in a polynomial time. In this model, the balancing amidst the exploration and exploitation capacities was enhanced by using a new pollination operator that was utilized to execute optimum scheduling with a lesser search complexity. Nonetheless, this work suffered with more VM migrations during the task scheduling process.

Kruekaew and Kimpan [21] developed a multi-objective artificial bee colony (ABC) with Q-learning (MOABCQ) to optimize the task scheduling in cloud. The incorporated Q-learning was a type of reinforcement learning which was used to enhance the operation speed of ABC. The MOABCQ was used for optimizing the schedule and resource usage for increasing the VM throughput and for ensuring the load balancing among the VMs by considering the cost, makespan, and resource usage. But, the developed MOABCQ was suitable for only the trained dataset, it did not consider the dynamic behavior of cloud environment.

Alsaidy et al. [22] presented the heuristic algorithms for assisting the Particle Swarm Optimization (PSO) in task scheduling process. A minimum completion time (MCT) and longest job to the fastest processor, aided for initializing the PSO's particles. The heuristic initialized MCT-PSO provided better load balancing in the cloud. This MCT-PSO was not efficient in the searching start points of VM and energy efficiency.

Saravanan et al. [23] developed the IWHOLF-TSC to perform the task scheduling over the cloud architecture. The IWHOLF integrated the principle of wild horse optimization (WHO) with the theory of levy flight. The IWHOLF-TSC developed the multi-objective fitness by minimizing the makespan and increasing the resource usage in cloud architecture. The developed IWHOLF-TSC did not consider the resource allocation and load balancing constraints of the cloud.

Azad et al. [24] presented the inverted ant colony optimisation (IACO) for solving the issue of task scheduling in cloud. In IACO, pheromone repellent was used instead of pheromone gravity, hence the impact of pheromone avoided the wrong selection. The FL with weight definition was utilized for monitoring the load balance and impact of pheromone repulsion in cloud. The developed IACO was used to enhance the load balancing and minimize the runtime. The increment in VMs was required to be analyzed for a better analysis of resource utilization over the cloud.

Mansouri et al. [25] developed the fuzzy system and modified particle swarm optimization (FMPSO) to perform task scheduling for improving the cloud throughput and load balancing. A modified velocity updating approach and roulette wheel selection were deployed for enhancing the global search. Here, the fuzzy system was used for computing the fitness where it used different inputs such as total execution time, RAM size, CPU speed and task length. This model lessened the resource usage and execution time, but the dynamic behavior of cloud was not considered by FMPSO while scheduling the tasks.

Mangalampalli et al. [26] presented the task-scheduling using whale optimization (TSWO) for cloud environment. The developed whale optimization computed the priorities and VMs for all tasks in the cloud to precisely allocate the tasks. The developed TSWO was analyzed only on the standard dataset, but it was not analyzed with the dynamic behavior of cloud.

Fu et al. [27] developed the PSO genetic hybrid algorithm according to phagocytosis namely PSO_PGA for scheduling the tasks in cloud. The PSO_PGA was modified the updating approach of location and speed in the conventional PSO. The fitness function (i.e., total completion time of task) and load balancing's standard deviation were considered for separating the population of every generation in the 1st and 2nd time respectively. The population of particle was always directed towards the optimum solution by using the feedback mechanism. However, an energy efficiency of the cloud was required to be considered for an effective task scheduling in cloud.

Zade and Mansouri [28] presented the game theory and fuzzy improved red fox optimizer (FIRFO) to schedule the tasks in the cloud. The search process of FIRFO was enhanced by using quasi-opposition based learning (QOBL), levy flight, cornu-spiral movement and fuzzy control systems. The QOBL was generated the initial population, Levy flight was improved the exploration capacity, fuzzy control systems was provided the balance among the exploration and exploitation, and the spiral movement was improved the local search ability. The developed FIRFO was required to be analyzed with the constraints of varying VMs.

Manikandan et al. [29] developed the hybrid whale optimization algorithm (WOA) based mutation-based Bees (MBA) to solve the multi-objective task scheduling issues in cloud. The hybrid WOA based MBA was concentrated in reducing the execution time and computational cost. The resource usage was maximized for reducing the makespan using the multiobjective behavior of hybrid WOA based MBA. The load balancing was required to be considered for further improving the performances.

The problems from the related work are given as follows: the related works mainly depended on the probabilistic approaches, however, they do not clearly address the essential uncertainty in the cloud environment using fuzzy logic and swarm intelligence approaches. Moreover, the existing models were not well responsive for dynamic variations in resource accessibility that caused the suboptimal task allocation and possibly ineffective resource usage. The solution given by the proposed research is as follows: the integration of FL with BFD supports the dynamic resource allocation according to the resource usage, power, cost and time. This adaptability is important in handling the varying conditions such as resource and workload variations which helps to perform an effective resource usage and VM allocation.

3 FL–BFD method

This research develops an FL-based BFD for effective task scheduling in cloud infrastructure. The data center gathers the tasks uploaded through the network to the cloud and FL–BFD allocates the tasks to adequate available resources based on the task's requirements and VM information. Here, the FL–BFD is operated by using resource usage, power, cost and time. The overall cloud architecture of task scheduling using FL–BFD is shown in Fig. 1.

3.1 System model

The cloud task scheduling is defined as the scheduling and assigning of different tasks to many Virtual Machines (VMs), and completing all the executed tasks in less execution period. The development of the load balancing approach involves task allocation in VMs and Physical Machines (PMs). The cloud has PMs which are represented as $Physical = \{Physical_1, Physical_2, \ldots, Physical_n\}; 1 \leq k \leq n$, where number of cloud PMs is denoted as n. PM contains sequential VMs represented as $Virtual = \left\{Virtual_1, Virtual_2, \ldots, Virtual_p\right\}; 1 \leq k \leq p$, where number of VMs is denoted as p.

The tasks are divided and organized into different chores considered as, $T = \{T_1, T_2, \ldots, T_f\}$ in a cloud computing model which are acquired from VMs. The loads existing in the VMs are balanced and stabilized, hence the cloud data center (CDC) works ordinarily in the cloud system. On the contrary, the underloaded and overloaded VMs are examined by using the load balancing algorithm. The chosen VMs (V_i) are balanced by implementing service-oriented restrictions such as memory, bandwidth, task migration cost and million instructions per second which are denoted in Eq. (1).

$$V_i = \{Np_i; N_i; BW_i; C_i; M_i\} \tag{1}$$

where number of processors are Np_i, number of MIPS are N_i, bandwidth utilization is BW_i, memory in terms of GB is M_i and task migration cost is C_i.

The parameters Np_i, N_i and M_i are ranged from 1 to a predefined constant c, whereas C_i and BW_i are ranged between $[0, 1]$. But the values of parameters cross their range when the VM is overloaded in the system. Further, the task allocated to VM is assigned to another VM which is underloaded when these parameters are smaller than the actual values. The remaining parameters considered for the calculation of tasks are execution time, task priority, and task communication costs that are shown in Eq. (2).

$$T_i = \{Priority_i; Execution_i; taskcommunicationcost_i\} \tag{2}$$

According to the task limitations, a high priority task and smaller operation time are key restrictions that are reassigned by overloaded tasks.

3.1.1 Incorporation of dependencies among tasks

At first, the system model has to be defined for denoting the different dependencies existing among the tasks such as data dependencies or task precedence for including the task dependencies in conventional workload model. This includes developing or extending the modern task class which comprises features or approaches for obtaining the data about the dependencies. Consequently, the task scheduling approaches in the CloudSim are required to be extended for considering these dependencies, by altering to prioritize the tasks with fulfilled dependencies or including the dependency restrictions into the scheduling decisions. Concurrently, the VM assignment policy using BFD is developed for accounting task dependencies in allocation. The simulation environment is developed for including the tasks with dependencies supporting the appropriate initialization of these dependencies. Further, the experimentations are developed for evaluating the impact of task dependencies via resource usage and completion time.

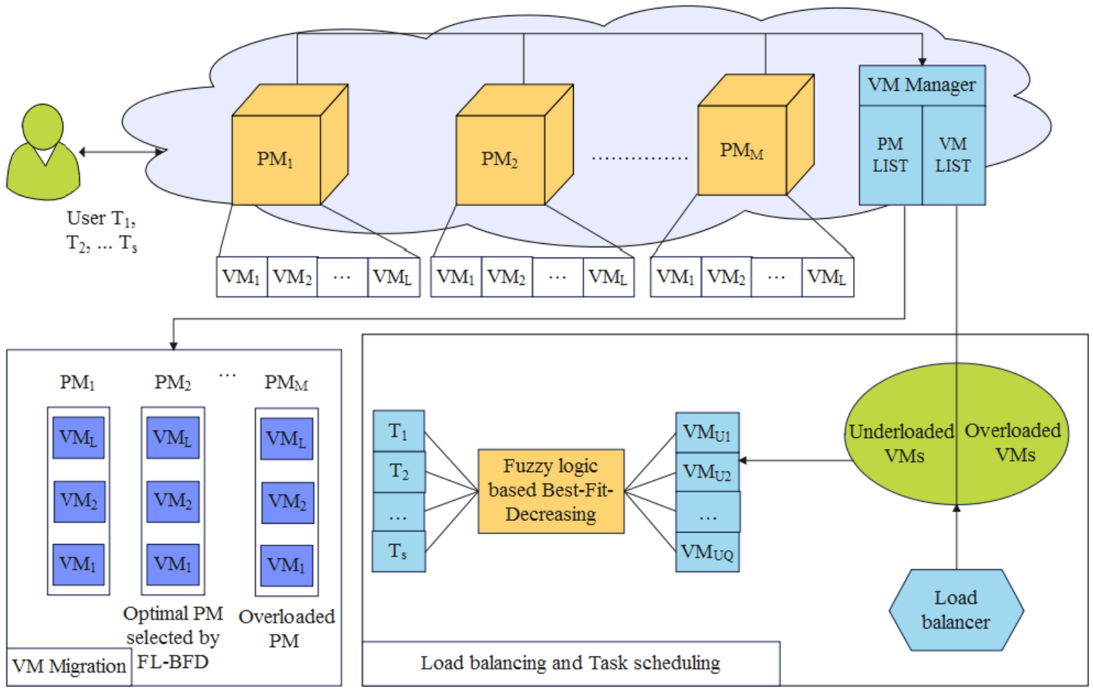

Fig. 1 Task scheduling using FL–BFD

Next, the simulation is modified for evaluating the results and recognizing how task dependencies impact the overall efficiency of cloud.

3.1.2 Incorporation of dependencies among VMs

An integration of BFD allocation with VM dependencies in CloudSim includes a systematic development and modification process. The model is defined to denote the VM dependencies, identifying the relationships which exist among the VMs such as the affinity or anti-affinity restrictions. Next, the VM in CloudSim is generated or extended including features and approaches for capturing the data related to VM dependencies. The BFD assignment policy is altered for considering the VM dependencies in the allocation that require the alterations in the sorting criteria for inspecting both the dependencies and resource requirements. The logic of host selection is altered as dependency-aware supporting that the selected host aligns with the dependencies of remaining VMs in the cloud. The principle to handle the VM dependencies is incorporated into the main assignment procedure of BFD. The CloudSim environment is updated for incorporating VM using the BFD. The simulation is altered for developing the VMs with dependencies, while assessments are done for evaluating the BFD.

3.2 FL with BFD for task scheduling

In this work, the utilization of fuzzy logic is proposed for combining the different objectives. The different objectives considered in this fuzzy logic are resource, power, time and cost. The fuzzy evaluation approach is employed for combining these four objectives into a single objective. The fuzzy searches for a better trade-off among these objectives on the basis of all other objectives noted during the optimization process. The FL incorporation enables the system for effectively handling the imprecise information and uncertainties. The existing task scheduling lacks the ability for adopting to dynamic and undefined cloud environments. FL plays a certain role in improving the load balancing approaches in the cloud. Here, the FL operates by considering the appropriate inputs such as resource usage, power, cost and time. These parameters define the fuzzy sets, divided into levels of low, medium and high via membership functions. The rules are employed in fuzzy input values for deriving the fuzzy output. Next, the defuzzification converts the fuzzy outputs into accurate choices, identifying the significant load balancing in the cloud. The outcome returns the migration of VM, altering resource assignments or remaining computations for optimizing the load of the system. The reliability of FL in acquiring and processing undefined data makes it specifically effective for load balancing choices in dynamic cloud. Moreover, the inclusion of feedback in FL makes it adaptive over time according to the varying cloud.

3.2.1 Fuzzy logic formulation for objective function

Fuzzy logic is considered for the VM Placement (VMP) issue, wherein it is mandatory for placing a group of VMs to minimize the power consumption and resource wastage. Here, it is required to discover an optimal solution based on resource utilization (R), power (P), cost (C) and time (T). The function vector ($\bar{\mathrm{F}}_{(x)}$) is expressed in Eq. (3).

$$\bar{\mathrm{F}}_{(x)} = (f_P(x), f_R(x), f_C(x), f_T(x)) \tag{3}$$

where power and resource used by the datacenter are denoted as $f_P(x)$ and $f_R(x)$, respectively, $f_C(x)$ denotes the cost, while $f_T(x)$ denotes the time.

The power consumption for active PMs ($P_{activePMs}$) is expressed in Eq. (4).

$$P_{activePMs} = \sum_{j=1}^{n} P_j(R(t)) \tag{4}$$

where n denotes the number of PMs. Equations (5) and (6) denote the resource utilization for active PMs ($R_{activePMs}$) in the cloud.

$$R_{activePMs} = \sum_{j=1}^{n} R_{pm_j} \tag{5}$$

$$R_{pm_j} = R_{pm_j^{cpu}} + R_{pm_j^{mem}} + R_{pm_j^{sto}} \tag{6}$$

where cpu, mem and sto denote CPU, memory and storage, respectively. The cost, as expressed in Eq. (7) denotes the total operation cost of the scheduling approach which is the sum of costs used by all task operations.

$$C = \sum_{j=1}^{p} vmC_j \tag{7}$$

Further, Eq. (8) expresses the job operation time on the VM.

$$T = \frac{tasksize}{vmspeed} \tag{8}$$

The membership functions for fuzzy subsets i.e., inputs of Power, Resource, Cost and Time are shown in Fig. 2. The linguistic variables of resource, power, cost and time are used for formulating the fuzzy logic. Each variable has three linguistic values which are, Low (L), Medium (M) and High (H). They are specified as follows:

- Resource: HNR, MNR and LNR.
- Power: HNP, MNP and LNP.
- Cost: HNC, MNC and LNC.
- Time: HNT, MNT and LNT.

Where, HNR, MNR and LNR denote higher, medium and smaller normalized resource usage, correspondingly, HNP, MNP and LNP denote higher, medium and smaller normalized power utilization, correspondingly, while HNC, MNC and LNC denote higher, medium and smaller normalized cost value, correspondingly, and HNT, MNT

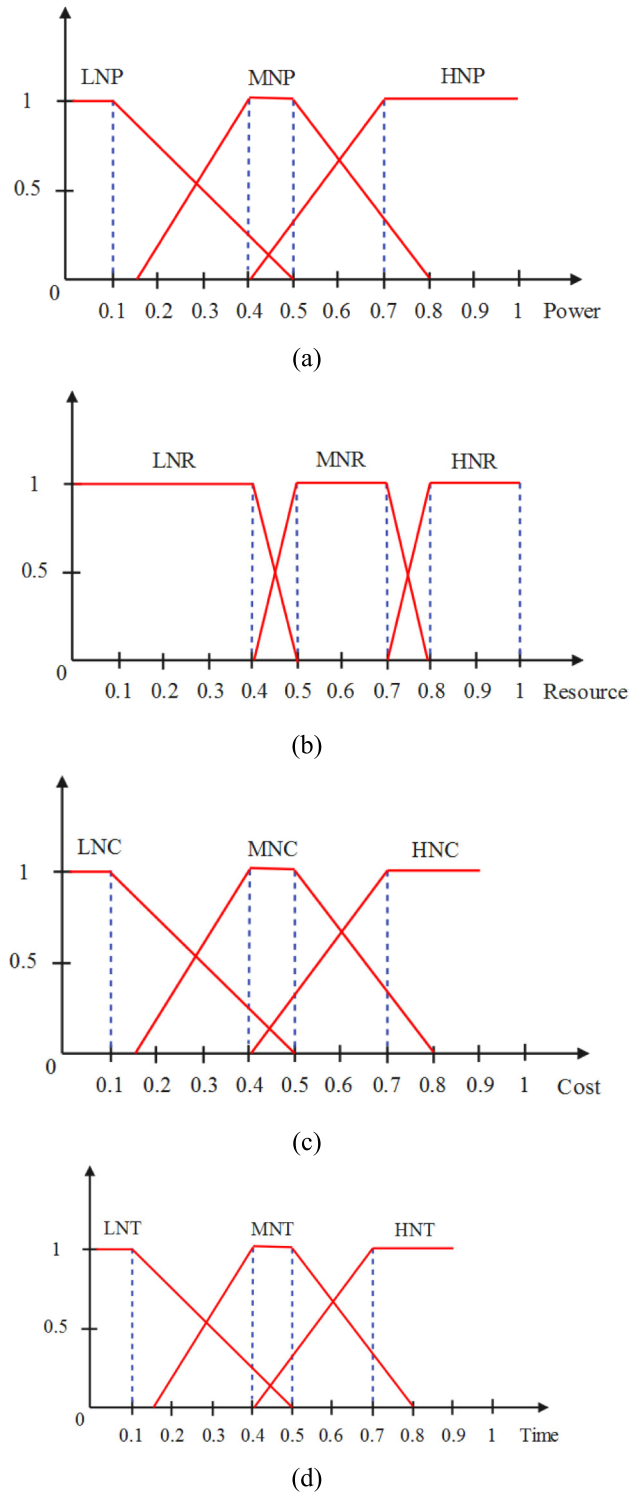

Fig. 2 Membership functions for fuzzy subsets: **a** power **b** resource **c** cost **d** time

and LNT denote higher, medium and smaller normalized time, correspondingly, in the cloud. Each linguistic value represents a fuzzy subset from the problem solution universe.

Further, the R, P, C and T are normalized between the range $[0, 1]$ for making the member function suitable for various issue instances. The normalized values P, R, C and T are computed using Eqs. (9) to (12).

$$NP = \frac{P_s}{P_{max}} \tag{9}$$

where the power usage for s^{th} solution is denoted as P_s and the maximum probable power usage of active PM is denoted as P_{max}.

$$NR = \frac{Usedresource}{Resourcecapacity} \tag{10}$$

where the utilized resources in the solution are denoted as $Usedresource$, while the resource capacity of the active PM is denoted as $Resourcecapacity$.

$$NC = \frac{C - C_{min}}{C_{max} - C_{min}} \tag{11}$$

$$NT = \frac{T - T_{min}}{T_{max} - T_{min}} \tag{12}$$

where NP, NR, NC and NT are correspondingly the normalized power, normalized resource, normalized cost and normalized time values which are given as input to the fuzzy logic.

3.2.2 Fuzzy logic rules

A highly wanted VMP solution is the one with a high membership at the fuzzy subcategories of HNR, LNP, LNC and LNT. In the fuzzy subset, one has to trade-off these distinct criteria against each other. The linguistic term is used to represent the trade-off of fuzzy subsets using one or more fuzzy logic rules.

The rules are generally "If…Then" declarations. Here, an "If" is the fuzzy base which is determined using fuzzy operators and linguistic values. The fuzzy AND is recognized using the function min according to the min–max logic, whereas the fuzzy OR is recognized using function max for combining 2 or more linguistic values. On the contrary, the "Then" is the subsequent. The fuzzy subset with a better outcome is discovered by using the linguistic value in the subsequent part. Therefore, an evaluating outcome of "If" part discovers the membership degree in the fuzzy subset with good placement outcomes based on the fuzzy rule.

The below fuzzy rule expresses the fuzzy subset with a good placement outcome.

R:1 $If((powerislow) AND(usageishigh))OR((powerismedium) AND(utilizationismedium))Thengoodsolution$

The rule 1 is examined using Eq. (13) using the min–max fuzzy logic.

$$\mu_{goodS}(x) = max(min(\mu_{LNP}(x), \mu_{HNR}(x), \mu_{LNT}(x), \mu_{LNC}(x)),$$
$$min(\mu_{MNP}(x), \mu_{MNR}(x), \mu_{MNC}(x), \mu_{MNT}(x)) \tag{13}$$

where the fuzzy's membership function of optimum outcomes is denoted as μ_{goodS} and μ defines the membership functions. The max and min operators are non-compensatory i.e., the weak components do not compensate for a stronger element. This criteria is unwanted for multi-objective optimization. Therefore, it is essential to utilize all data sources. In this scenario, it is suggested to use a fuzzy AND and OR operator that assists all objective values to create an impact on the operator's outcome. Further, a weighted averaging operator is incorporated for relaxing the rule aggregation. Hence, the rule 1 of fuzzy logic i.e., R:1 is estimated as shown in Eqs. (14) to (18).

$$\mu_{goodS}(x) = \beta \times \max(\mu_1, \mu_2, \mu_3, \mu_4) + (1 - \beta) \times \frac{1}{2}\sum_{i=1}^{4} \mu_i(x) \tag{14}$$

$$\mu_1(x) = \beta \times \min(\mu_{LNP}(x), \mu_{HNR}(x)) + (1 - \beta) \times \frac{1}{2}(\mu_{LNP}(x) + \mu_{HNR}(x)) \tag{15}$$

$$\mu_2(x) = \beta \times \min(\mu_{MNP}(x), \mu_{MNR}(x)) + (1 - \beta) \times \frac{1}{2}(\mu_{MNP}(x) + \mu_{MNR}(x)) \tag{16}$$

$$\mu_3(x) = \beta \times \min(\mu_{LNC}(x), \mu_{MNC}(x), \mu_{HNC}(x)) + (1 - \beta) \times \frac{1}{2}(\mu_{LNC}(x) + \mu_{MNC}(x) + \mu_{HNC}(x)) \tag{17}$$

$$\mu_3(x) = \beta \times \min(\mu_{LNT}(x), \mu_{MNT}(x), \mu_{HNT}(x)) + (1 - \beta) \times \frac{1}{2}(\mu_{LNT}(x) + \mu_{MNT}(x) + \mu_{HNT}(x)) \tag{18}$$

where β is a value in the range of $[0, 1]$.

3.3 BFD-based scheduling

The Best-Fit-Decreasing (BFD) algorithm is used for solving the issue of VM reallocation. The fuzzy logic is used in the cloud architecture for reallocating the VMs, and Fig. 3 shows the flowchart for VM reallocation using FL–BFD. The BFD designates the VMs' operation from demand submission to the execution. According to the time-driven simulation, the received VMs are maintained

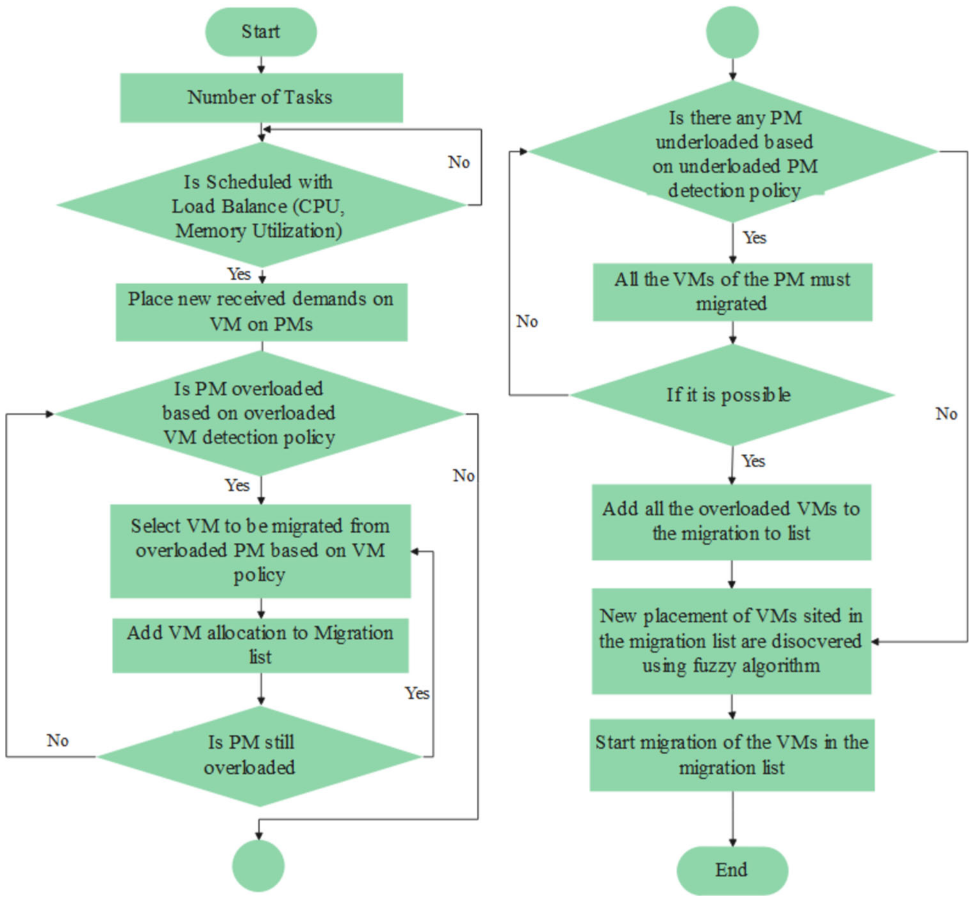
Start
Number of Tasks
Is Scheduled with Load Balance (CPU, Memory Utilization)
No
Yes
Place new received demands on VM on PMs
Is PM overloaded based on overloaded VM detection policy
Yes
No
Select VM to be migrated from overloaded PM based on VM policy
Add VM allocation to Migration list
Yes
No
Is PM still overloaded
Is there any PM underloaded based on underloaded PM detection policy
Yes
All the VMs of the PM must migrated
No
If it is possible
No
Yes
Add all the overloaded VMs to the migration to list
New placement of VMs sited in the migration list are disocvered using fuzzy algorithm
Start migration of the VMs in the migration list
End

Fig. 3 Flowchart for VM reallocation using FL–BFD

in the cloud. The VMs are sorted in descending order of CPU usage, where each VM is assigned to the PM as the cloud receives new VMs. Each PM state is controlled for avoiding the over/underloaded conditions, in order to frequently optimize the current assignment of VMs. Some VMs are required to move from PM as per the selection criteria, and the chosen VMs are included in the relocation list when the overloaded PM is found in the cloud. All VMs are moved from the PM and included in the relocation list when an underloaded PM is found in the cloud. Next, the fuzzy-based BFD is used to reallocate the chosen VMs to another PM other than the under/overloaded PMs in the VMs' list. The VMs are arranged in descending order using BFD and are allocated to the PM. The VM is positioned in the PM with least existing resource capacity which assists the resources needed by the VM. A new PM is ON when the active PM does not assist the VM. Similarly, a set of VMs is placed on the PM based on the fuzzy logic. In this work, the objectives: power, resource, cost and time, are maximized in a fuzzy way for performing an effective VM reallocation. For the selection of an appropriate VM in a decreasing order of resource necessities, the BFD reduces the wasted resources and improves the overall system efficiency. This leads to making the BFE suitable for dynamic workloads, as it adaptively varies according to the demands by allocating the VMs.

In the domain of load balancing with FL in cloud, the roles of the VM manager and migration approaches are crucial for the effectiveness of the overall system efficiency. The VM manager understands the information about load offered by the BFD and transforms into an effective decision in the cloud. The BFD performs the allocation, de-allocation and migration of VM over hosts for maintaining an appropriate resource usage. Simultaneously, migration acts as an operation of load balancing approach dynamically relocating VMs according to the BFD outputs. BFD leads the decision of VM migration for addressing the resource utilization imbalance in the resource i.e., uneven workload dissemination and higher CPU usage. The combination of VM manager and migration generate a combined solution which fine tunes the awareness of BFD for adaptively maintaining the VM assignments and migrations. This helps in effective load balancing, alongside maximizes the system efficiency in the cloud.

The algorithm for FL–BFD based task scheduling in cloud is given below:

Input: TaskList, ResourceList, FL

Output: Task assignment in cloud

1. Organize the tasks in decreasing order according to the sizes (BFD).
2. for every task in TaskList:
3. Use FL for identifying the degree of suitability for every resource.
4. Choose the resource with the higher degree of membership.
5. End for
6. Assign the chosen task for the respective selected resource.
7. The resource's availability is updated according to the size of assigned task.
8. Repeat steps 2–7 until all tasks are assigned.

4 Results and discussion

In this work, the Java with JFuzzyLogic and CloudSim simulators are used to design and simulate the FL–BFD-based task scheduling in cloud infrastructure. JFuzzyLogic is generally a Java library used to design and simulate the fuzzy logic systems. Subsequently, the JFuzzyLogic is integrated with CloudSim by designing the FL controllers and decision-making modules in Java, thereby integrating the FL with CloudSim. The system configuration used for this work is an i5 processor, 8 GB RAM and Windows 10 operating system. The developed research considers 40–200 VMs and 200–1000 tasks. The specifications of the cloud are given in Table 1. Makespan, execution time, degree of imbalance, energy consumption and SLA violations are the important metrics used for analyzing the FL–BFD method. The degree of imbalance is used to offer a value for assessing how unequally the resources are disseminated over processing elements in the cloud. The SLA violation is mainly based on the host's active time and performance degradation.

Table 1 Specifications of cloud

Parameters	Values
Number of VMs	40–200
Number of tasks	200–1000
Size of tasks (MI)	1000–4000
Power consumption of VMs (Watt)	200–1000
Power consumption percentage for idle to active state	0.6–0.7
VM execution rate (MIPS)	1000–5000

4.1 Performance evaluation

In this section, the FL–BFD method is analyzed for two different scenarios which are specified in the below Table 2. Here, the FL–BFD is analyzed for varying tasks and VMs, in the scenarios 1 and 2, respectively. From the Table 2, it is represented that the scalability of the FL–BFD is analyzed by two different configurations: (1) fixed number of VMs and different number of tasks, and (2) different number of VMs and fixed number of tasks.

In scenario 1, the amount of VMs is fixed, and tasks are varied from 200 to 1000 for analyzing the scheduling performances. Table 3 shows the analysis of FL–BFD with conventional fuzzy for scenario 1. Figures 4, 5, 6, 7 and 8 show the comparison of FL–BFD and fuzzy for makespan, execution time, degree of imbalance, energy consumption, and SLA violations, correspondingly. These figures represents that the increasing the number of tasks causes the increment in amount of resources. From the results, it is evident that the FL–BFD achieves better performance when compared to the Fuzzy, even with the increment in number of tasks. The FL–BFD has the better scalability than the fuzzy for both the less and high amount of tasks. For example, the makespan of FL–BFD is 9.2 ms, whereas the Fuzzy obtains 12.6 ms. It is to be noted that the existing fuzzy approach fails to handle the dynamic and uncertain situations of cloud, therefore it obtains a lesser performance. SLA represents the terms and conditions between the service providers and consumers, evaluating the desired levels of availability, service quality and performance. On the contrary, energy efficiency denotes the optimization of energy consumption and resource usage. SLA denotes the resources assigned to service and these resources influence the energy utilization. The energy consumed for 1000 tasks is 567 mJ, while the SLA violations obtained are 12. The FL–BFD enables an effective VM reallocation according to the user requirements along with the resource usage, power, cost and time. Hence, the FL–BFD optimizes the resource usage based on the adaptability of FL and efficiency of BFD, thereby minimizing the resource usage and execution time. Moreover, the allocation of VM based on the task's resource requirements facilitates in minimizing the execution time that enhances the system efficiency and its responsiveness.

Table 2 Scenario information

Parameters	Scenario 1	Scenario 2
Number of VMs	100	40–200
Number of tasks	200–1000	500

Table 3 Analysis of FL–BFD for scenario 1

Performances	Tasks	Fuzzy	FL–BFD
Makespan (ms)	200	2.8	1.8
	400	3.8	2.4
	600	6.6	4.8
	800	10.8	8.5
	1000	12.6	9.2
Execution time (ms)	200	145.8	132.5
	400	262.6	216.8
	600	445.4	386.6
	800	578.6	450.5
	1000	689.5	465.7
Degree of imbalance	200	0.6	0.4
	400	0.6	0.4
	600	0.9	0.8
	800	1.0	0.8
	1000	1.2	1.0
Energy consumption (mJ)	200	118	102
	400	221	212
	600	232	220
	800	436	365
	1000	678	567
SLA violations	200	7	3
	400	8	7
	600	10	8
	800	13	11
	1000	16	12

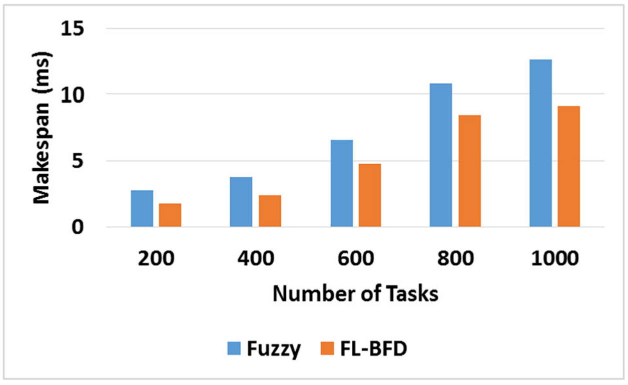

Fig. 4 Graph of makespan for scenario 1

The number of tasks are kept as constant, taken as 500, and the number of VMs are varied from 40 to 200 for analyzing the load balancing. Table 4 shows the analysis of FL–BFD with conventional fuzzy for scenario 2. The comparison of FL–BFD and fuzzy for makespan, execution time, degree of imbalance, energy consumption, and SLA violations for the 2nd scenario is shown in Figs. 9, 10, 11,

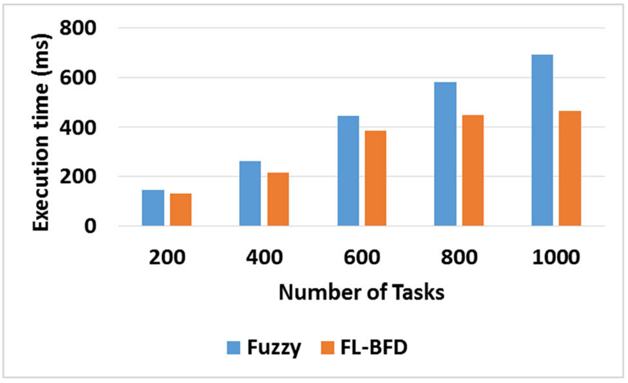

Fig. 5 Graph of execution time for scenario 1

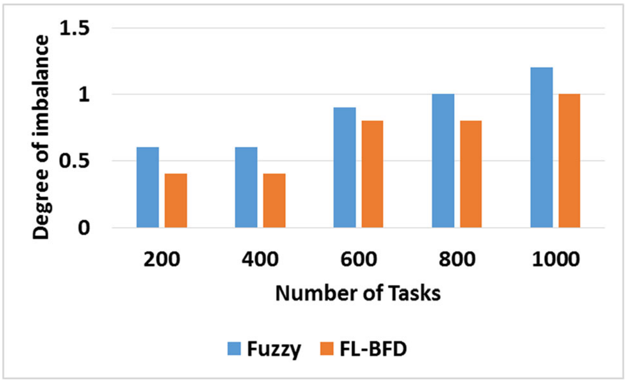

Fig. 6 Graph of degree of imbalance for scenario 1

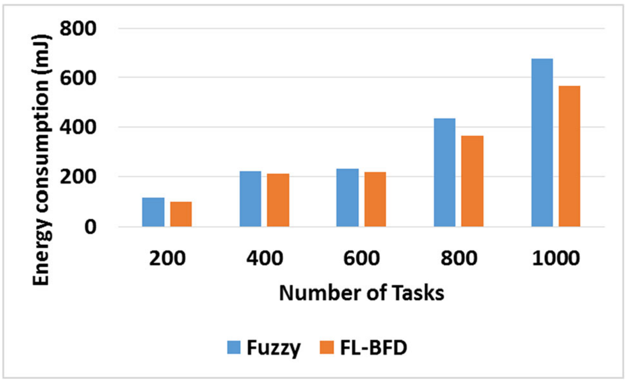

Fig. 7 Graph of energy consumption for scenario 1

12 and 13, respectively. The increment in the number of VMs increases the amount of resources. This analysis shows that the FL–BFD achieves better performance when compared to the Fuzzy, even when the cloud is analyzed with varying number of VMs. Therefore, the FL–BFD has better scalability than the fuzzy for both the less and high amount of VMs. For example, the makespan of FL–BFD with 200 VMs is 1.2 ms, whereas the Fuzzy obtains 1.8 ms.

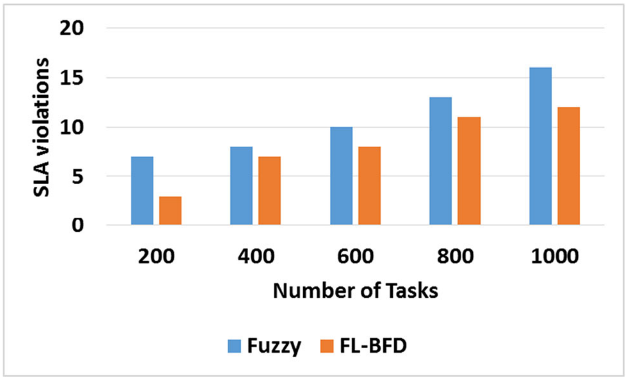

Fig. 8 Graph of SLA violations for scenario 1

Table 4 Analysis of FL–BFD for scenario 2

Performances	Number of VMs	Fuzzy	FL–BFD
Makespan (ms)	40	18.9	16.4
	80	4.8	3.8
	120	4.5	4.2
	160	1.9	1.6
	200	1.8	1.2
Execution time (ms)	40	337.2	237.9
	80	456.0	318.5
	120	328.0	312.4
	160	398.8	304.8
	200	400.0	389.5
Degree of imbalance	40	0.8	0.6
	80	0.5	0.2
	120	0.5	0.2
	160	0.6	0.4
	200	0.7	0.5
Energy consumption (mJ)	40	350.0	325.5
	80	256.8	235.0
	120	189.0	156.5
	160	218.9	203.2
	200	202.1	182.2
SLA violations	40	2	1
	80	2	1
	120	4	1
	160	4	2
	200	4	2

Further, the scenario 2 analyzed in the CloudSim is also analyzed using Amazon Web Services (AWS) for a real time analysis which is provided in Table 5. This analysis is used to provide the FL–BFD performance when it is processed with real time data center, PM and cloud environment. Due to real time internet and host devices, the FL–BFD results in marginal decrease in performances when compared to the CloudSim simulation.

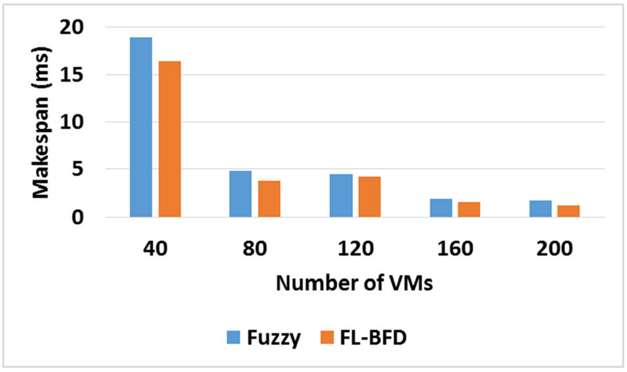

Fig. 9 Graph of makespan for scenario 2

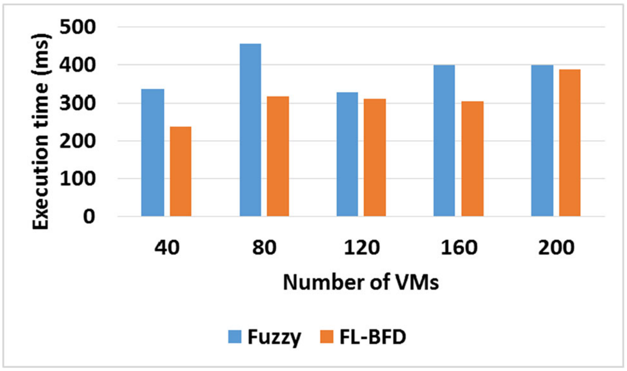

Fig. 10 Graph of execution time for scenario 2

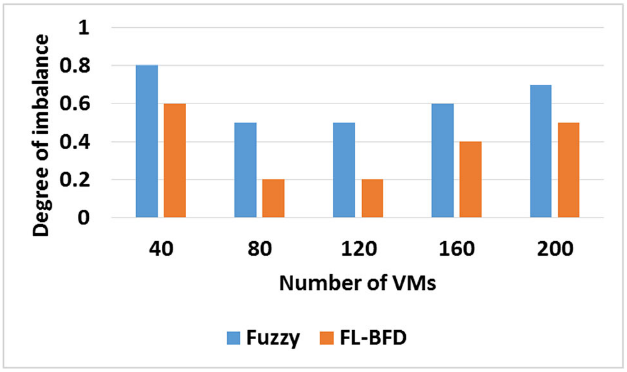

Fig. 11 Graph of degree of imbalance for scenario 2

4.2 Comparative analysis

This section provides a comparison between FL–BFD and the existing researches which are, IWHOLF-TSC [22], MCT-PSO [23], IACO [24], FMPSO [25] and TSWO [26]. Here as well, the comparison is made for five different scenarios. The information about scenarios 3, 4 and 5 are provided in the Table 6. The TSWO [26] is designed for

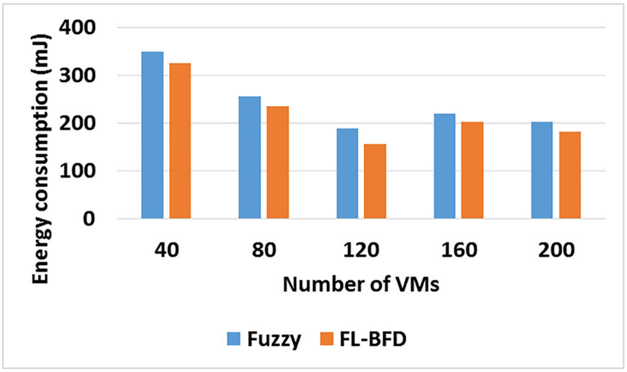

Fig. 12 Graph of energy consumption for scenario 2

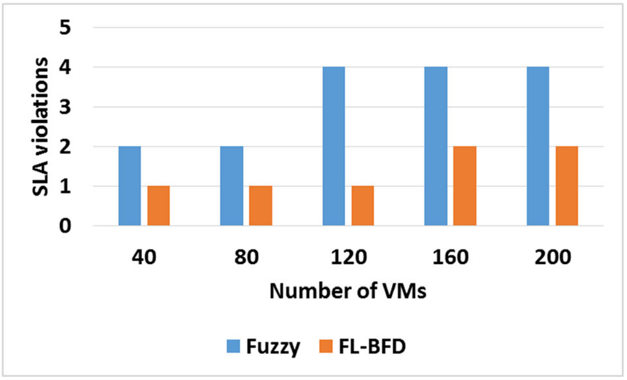

Fig. 13 Graph of SLA violations for scenario 2

Table 5 FL–BFD evaluation with AWS for scenario 2

Performances	Number of VMs	Fuzzy	FL–BFD
Makespan (ms)	40	19.722	17.013
	80	5.780	4.717
	120	5.363	5.037
	160	2.463	2.318
	200	2.264	1.848
Execution time (ms)	40	338.145	238.582
	80	456.191	318.709
	120	328.632	313.308
	160	399.559	305.402
	200	400.267	390.043
Degree of imbalance	40	0.804	1.340
	80	0.507	0.209
	120	0.507	0.210
	160	0.600	0.400
	200	0.702	0.507
Energy consumption (mJ)	40	350.354	326.483
	80	257.092	235.420
	120	189.412	157.453
	160	219.633	203.997
	200	203.029	182.656
SLA violations	40	2.000	1.000
	80	2.000	1.000
	120	5.000	1.000
	160	5.000	2.000
	200	5.000	2.000

Table 6 Information about scenario 3, 4 and 5

Parameters	Scenario 3	Scenario 4	Scenario 5
Number of VMs	50	40	20
Number of tasks	50	40	100–1000

the same specifications of power consumption and VM execution rate, as mentioned in the Table 1. Tables 7, 8, 9, 10, 11 offer the comparison correspondingly for the scenarios 1, 2, 3, 4 and 5, where NA denotes values not available in the respective research. From this analysis, it is determined that the FL–BFD outperforms the IWHOLF-TSC [22], MCT-PSO [23], IACO [24], FMPSO [25] and TSWO [26]. For example, the makespan of FL–BFD with 1000 tasks is 9.2 ms, whereas that of the IWHOLF-TSC [22] is 776 ms, and that of MCT-PSO [23] is 17.6 ms. Moreover, the SLA violation comparisons are provided in Table 11, where the assessment is done under random generated workload and BigDataBench workload. This evaluation concludes that FL–BFD has lesser SLA violations than the TSWO [26]. The FL's adaptability and efficiency of BFD is availed to optimize resource usage. The consideration of the dynamic behavior of cloud environments with FL–BFD is profited to minimize the resource usage and execution time. This adaptability is important in handling the varying conditions of resource and workload variations that help to perform effective resource usage and VM allocation. The developed FL–BFD is useful in mobile applications, cloud analytics and AWS services.

Table 7 Comparison of FL–BFD for scenario 1

Performances	Tasks	IWHOLF-TSC [22]	MCT-PSO [23]	FL–BFD
Makespan (ms)	200	96	2.2	1.8
	400	265	4.2	2.4
	600	480	8.5	4.8
	800	659	12.1	8.5
	1000	776	17.6	9.2
Execution time (ms)	200	42.12	155.8	132.5
	400	210.72	326.7	216.8
	600	370.45	535.8	386.6
	800	464.52	716.5	450.5
	1000	489.35	972.3	465.7
Degree of imbalance	200	0.866	0.9	0.4
	400	0.852	0.6	0.4
	600	0.871	1.0	0.8
	800	0.857	1.1	0.8
	1000	0.885	1.2	1.0
Energy consumption (mJ)	200	NA	122	102
	400	NA	242	212
	600	NA	440	220
	800	NA	599	365
	1000	NA	822	567

Table 8 Comparison of FL–BFD for scenario 2

Performances	Number of VMs	MCT-PSO [23]	FL–BFD
Makespan (ms)	40	25.9	16.4
	80	9.7	3.8
	120	4.9	4.2
	160	3.7	1.6
	200	2.7	1.2
Execution time (ms)	40	486.8	237.9
	80	518.7	318.5
	120	427.0	312.4
	160	417.6	304.8
	200	410.1	389.5
Degree of imbalance	40	1.0	0.6
	80	0.8	0.2
	120	0.7	0.2
	160	0.7	0.4
	200	0.7	0.5
Energy consumption (mJ)	40	362	325.5
	80	365	235.0
	120	311	156.5
	160	318	203.2
	200	304	182.2

Table 9 Comparison of FL–BFD for scenario 3

Number of tasks	Execution time (ms)	
	IACO [24]	FL–BFD
250	1,251,635.3	1,023,213.45
300	1,501,962.34	1,437,837.34
350	1,752,289.38	1,574,293.24
400	2,002,616.42	1,875,392.92

Table 10 Comparison of FL–BFD for scenario 4

Number of tasks	Execution time (ms)	
	FMPSO [25]	FL–BFD
100	500	431
200	1245	850
300	1534	1290
400	2034	1879
500	2120	1982
600	3408	2805
700	4205	3792

5 Conclusion

Cloud computing is a modern concept deployed in the IT systems, providing various advantages including enhanced response time, computing power and cost minimization. The users possess full advantages of cloud services to satisfy their requirements. This research specifically concentrates on task scheduling according to the user's demand. In this research, the FL–BFD is developed for an effective reallocation of VM in cloud, as per the user's demand. The task scheduling via VM reallocation is carried out by using different factors that are, resource usage, power, cost and time. The BFD decides the operation of VM, right from demand submission to the execution. Moreover, each PM state is handled to eliminate the over/under-loaded conditions and frequently optimize the current assignment of VMs. From the analysis, it is evident that the developed FL–BFD provides better performance than the IWHOLF-TSC, MCT-PSO, IACO, FMPSO, and TSWO. The makespan of FL–BFD with 1000 tasks is 9.2 ms which is superior in contrast to IWHOLF-TSC and MCT-PSO. In the future, the conditions of FL can be improved for further enhancing the energy efficiency of cloud environment.

Table 11 Comparison of FL–BFD for scenario 5

Workload	Number of VMs	SLA violations	
		TSWO [26]	FL–BFD
Random generated workload	100	8	6
	500	9	8
	1000	18	10
BigDataBench workload	100	8	6
	500	9	8
	1000	12	10

Author contributions Nitin Thapliyal: conceptualization; investigation; methodology; visualization; formal analysis; writing—original draft. Priti Dimri: data curation; resources; validation; supervision; writing—review and editing; project administration. All authors have read and approved the final manuscript.

Funding This research received no external funding.

Data availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethics approval I/We declare that the work submitted for publication is original, previously unpublished in English or any other language(s), and not under consideration for publication elsewhere.

Consent for publication I certify that all the authors have approved the paper for release and are in agreement with its content.

Consent to participate Not applicable.

Informed consent Not applicable.

References

1. Siddesha, K., Jayaramaiah, G.V., Singh, C.: A novel deep reinforcement learning scheme for task scheduling in cloud computing. Cluster Comput. **25**(6), 4171–4188 (2022). https://doi.org/10.1007/s10586-022-03630-2
2. Khan, M.S.A., Santhosh, R.: Task scheduling in cloud computing using hybrid optimization algorithm. Soft. Comput.Comput. **26**(23), 13069–13079 (2022). https://doi.org/10.1007/s00500-021-06488-5
3. Mangalampalli, S., Swain, S.K., Mangalampalli, V.K.: Multi objective task scheduling in cloud computing using cat swarm optimization algorithm. Arab. J. Sci. Eng. **47**(2), 1821–1830 (2022). https://doi.org/10.1007/s13369-021-06076-7
4. Imene, L., Sihem, S., Okba, K., Mohamed, B.: A third generation genetic algorithm NSGAIII for task scheduling in cloud computing. J. King Saud Univ. Comput. Inf. Sci. **34**(9), 7515–7529 (2022). https://doi.org/10.1016/j.jksuci.2022.03.017
5. Sharma, M., Kumar, M., Samriya, J.K.: An optimistic approach for task scheduling in cloud computing. Int. J. Inf. Technol. **14**(6), 2951–2961 (2022). https://doi.org/10.1007/s41870-022-01045-1
6. Abualigah, L., Alkhrabsheh, M.: Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. J. Supercomput.Supercomput. **78**(1), 740–765 (2022). https://doi.org/10.1007/s11227-021-03915-0
7. Praveen, S.P., Ghasempoor, H., Shahabi, N., Izanloo, F.: A hybrid gravitational emulation local search-based algorithm for task scheduling in cloud computing. Math. Probl. Eng.Probl. Eng. **2023**, 6516482 (2023). https://doi.org/10.1155/2023/6516482
8. Sharma, N., Sonal, Garg, P.: Ant colony based optimization model for QoS-based task scheduling in cloud computing environment. Meas. Sens. **24**, 100531 (2022). https://doi.org/10.1016/j.measen.2022.100531
9. Panda, S.K., Nanda, S.S., Bhoi, S.K.: A pair-based task scheduling algorithm for cloud computing environment. J. King Saud Univ. Comput. Inf. Sci. **34**(1), 1434–1445 (2022). https://doi.org/10.1016/j.jksuci.2018.10.001
10. Nayak, S.C., Parida, S., Tripathy, C., Pattnaik, P.K.: An enhanced deadline constraint based task scheduling mechanism for cloud environment. J. King Saud Univ. Comput. Inf. Sci. **34**(2), 282–294 (2022). https://doi.org/10.1016/j.jksuci.2018.10.009
11. Kang, K., Ding, D., Xie, H., Yin, Q., Zeng, J.: Adaptive DRL-based task scheduling for energy-efficient cloud computing. IEEE Trans. Netw. Serv. Manag.Netw. Serv. Manag. **19**(4), 4948–4961 (2022). https://doi.org/10.1109/TNSM.2021.3137926
12. Gupta, P., Rawat, P.S., Saini, D.K., Vidyarthi, A., Alharbi, M.: Neural network inspired differential evolution based task scheduling for cloud infrastructure. Alex. Eng. J. **73**, 217–230 (2023). https://doi.org/10.1016/j.aej.2023.04.032
13. Gupta, S., Iyer, S., Agarwal, G., Manoharan, P., Algarni, A.D., Aldehim, G., Raahemifar, K.: Efficient prioritization and processor selection schemes for HEFT algorithm: a makespan optimizer for task scheduling in cloud environment. Electronics **11**(16), 2557 (2022). https://doi.org/10.3390/electronics11162557
14. Mahmoud, H., Thabet, M., Khafagy, M.H., Omara, F.A.: Multiobjective task scheduling in cloud environment using decision tree algorithm. IEEE Access **10**, 36140–36151 (2022). https://doi.org/10.1109/ACCESS.2022.3163273
15 Pirozmand, P., Javadpour, A., Nazarian, H., Pinto, P., Mirkamali, S., Ja'fari, F.: GSAGA: a hybrid algorithm for task scheduling in cloud infrastructure. J. Supercomput.Supercomput. **78**(15), 17423–17449 (2022). https://doi.org/10.1007/s11227-022-04539-8
16. Ghafari, R., Mansouri, N.: Improved Harris hawks optimizer with chaotic maps and opposition-based learning for task scheduling in cloud environment. Cluster Comput. (2023). https://doi.org/10.1007/s10586-023-04021-x
17. Malathi, K., Priyadarsini, K.: Hybrid lion–GA optimization algorithm-based task scheduling approach in cloud computing. Appl. Nanosci.Nanosci. **13**(3), 2601–2610 (2023). https://doi.org/10.1007/s13204-021-02336-y
18. Lipsa, S., Dash, R.K., Ivković, N., Cengiz, K.: Task scheduling in cloud computing: a priority-based heuristic approach. IEEE Access **11**, 27111–27126 (2023). https://doi.org/10.1109/ACCESS.2023.3255781
19. Saroit, I.A., Tarek, D.: LBCC-Hung: a load balancing protocol for cloud computing based on Hungarian method. Egypt. Inf. J. **24**(3), 100387 (2023). https://doi.org/10.1016/j.eij.2023.100387
20. Emami, H.: Cloud task scheduling using enhanced sunflower optimization algorithm. ICT Express **8**(1), 97–100 (2022). https://doi.org/10.1016/j.icte.2021.08.001
21. Kruekaew, B., Kimpan, W.: 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 (2022). https://doi.org/10.1109/ACCESS.2022.3149955
22. Alsaidy, S.A., Abbood, A.D., Sahib, M.A.: Heuristic initialization of PSO task scheduling algorithm in cloud computing. J. King Saud Univ. Comput. Inf. Sci. **34**(6A), 2370–2382 (2022). https://doi.org/10.1016/j.jksuci.2020.11.002
23. Saravanan, G., Neelakandan, S., Ezhumalai, P., Maurya, S.: Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing. J. Cloud Comput. **12**, 24 (2023). https://doi.org/10.1186/s13677-023-00401-1
24. Azad, P., Navimipour, N.J., Hosseinzadeh, M.: A fuzzy-based method for task scheduling in the cloud environments using inverted ant colony optimisation algorithm. Int. J. Bio-Inspired Comput. **14**(2), 125–137 (2019). https://doi.org/10.1504/IJBIC.2019.101638
25. Mansouri, N., Zade, B.M.H., Javidi, M.M.: Hybrid task scheduling strategy for cloud computing by modified particle

swarm optimization and fuzzy theory. Comput. Ind. Eng.. Ind. Eng. **130**, 597–633 (2019). https://doi.org/10.1016/j.cie.2019.03.006
26. Mangalampalli, S., Swain, S.K., Karri, G.R., Mishra, S.: SLA aware task-scheduling algorithm in cloud computing using whale optimization algorithm. Sci. Program. **2023**, 8830895 (2023). https://doi.org/10.1155/2023/8830895
27. Fu, X., Sun, Y., Wang, H., Li, H.: Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm. Clust. Comput.. Comput. **26**(5), 2479–2488 (2023)
28. Zade, B.M.H., Mansouri, N.: Improved red fox optimizer with fuzzy theory and game theory for task scheduling in cloud environment. J. Comput. Sci.Comput. Sci. **63**, 101805 (2022)
29. Manikandan, N., Gobalakrishnan, N., Pradeep, K.: Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment. Comput. Commun.. Commun. **187**, 35–44 (2022)

Nitin Thapliyal has been a dedicated professional in the field of Computer Science and Engineering. With a Master's degree (M.Tech.) in CSE and a Ph.D. in pursuit, he had brought a rich educational background to the table. Over the span of 14 years, he had immersed himself in various aspects of the tech world, gaining extensive experience that had honed his skills and knowledge. His passion had lain at the intersection of cutting-edge technologies, and his research areas had reflected this enthusiasm. His specialization and area of interest includes in Data analytics, cloud Computing, Machine Learning, Databases, and Linux systems. These domains had not only captivated his academic pursuits but had also driven his practical endeavours. He Has published more than 15 research papers/articles and presented in national conferences. As he had continued to explore the realms of academia, industry, and innovation, he had been excited to see how his experiences and expertise would further evolve. His goal had been to remain at the forefront of technological advancements, leveraging his skills to make meaningful contributions that had positively impacted the digital world.

Priti Dimri is a Professor (Department of Computer Science and Applications) at GBPEC ghurdwari uttarakhand. Her professional qualification includes Ph.D. M.C.A., P.G.D.C.A., M.B.A. (Finance), ITIL V3.0 Foundation. She has total of more than 17 Years in Industry, Research, Teaching & Training and is working in green computing, DBMS, cloud computing area.