Dynamic Resource Allocation Using Improved Firefly Optimization Algorithm in Cloud Environment

Simin Abedi, Mostafa Ghobaei-Arani, Ehsan Khorami & Musa Mojarad

To cite this article: Simin Abedi, Mostafa Ghobaei-Arani, Ehsan Khorami & Musa Mojarad (2022) Dynamic Resource Allocation Using Improved Firefly Optimization Algorithm in Cloud Environment, Applied Artificial Intelligence, 36:1, 2055394, DOI: 10.1080/08839514.2022.2055394

To link to this article: https://doi.org/10.1080/08839514.2022.2055394

APPLIED ARTIFICIAL INTELLIGENCE
2022, VOL. 36, NO. 1, e2055394 (2627 pages)
https://doi.org/10.1080/08839514.2022.2055394

Dynamic Resource Allocation Using Improved Firefly Optimization Algorithm in Cloud Environment

Simin Abedi[a], Mostafa Ghobaei-Arani[b], Ehsan Khorami[c], and Musa Mojarad[d]

[a]Department of Computer Engineering, Mahallat Branch, Islamic Azad University, Mahallat, Iran; [b]Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran; [c]Department of Computer Engineering, Kermanshah Branch, Islamic Azad University, Kermanshah, Iran; [d]Department of Computer Engineering, Firoozabad Branch, Islamic Azad University, Firoozabad, Iran

ABSTRACT
Today, cloud computing has provided a suitable platform to meet the computing needs of users. One of the most important challenges facing cloud computing is Dynamic Resource Allocation (DSA), which is in the NP-Hard class. One of the goals of the DSA is to utilization resources efficiently and maximize productivity. In this paper, an improved Firefly algorithm based on load balancing optimization is introduced to solve the DSA problem called IFA-DSA. In addition to balancing workloads between existing virtual machines, IFA-DSA also reduces completion time by selecting appropriate objectives in the fitness function. The best sequence of tasks for resource allocation is formulated as a multi-objective problem. The intended objectives are load balancing, completion time, average runtime, and migration rate. In order to improve the initial population creation in the firefly algorithm, a heuristic method is used instead of a random approach. In the heuristic method, the initial population is created based on the priority of tasks, where the priority of each task is determined based on the pay as you use model and a fuzzy approach. The results of the experiments show the superiority of the proposed method in the makespan criterion over the ICFA method by an average of 3%.

Introduction

Cloud computing is a general term for anything that includes the provision of internet-hosted services (Naha et al. 2020). For example, such as digital marketing and e-mail marketing (Pawar and Wagh 2012). In addition, cloud computing can be introduced as providing computer services such as storage, database, software, networking and analytics that provide more flexible resources (Rezaeipanah, Mojarad, and Fakhari 2022). In this technology, the user does not have access to technical details and only sees its appearance (Liu et al. 2022). The logic behind cloud computing is time sharing. In other words, different computer resources are shared between multiple users using multi-programming and multi-tasking mechanisms. This approach was first used in

CONTACT Simin Abedi siminabedi68@yahoo.com; Mostafa Ghobaei-Arani m.ghobaei@qom-iau.ac.ir Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran

the 1950s (Berahmand et al. 2021). At this time, several users shared its services with access to a central computer, because of the high price and large size of central computers, it was not possible to provide an independent system for each user. Thus, cloud services can be considered as a way to share computers in the 1950s (Wang et al. 2015).

Internet technology is growing rapidly and its use is widespread, and in the meantime cloud computing is emerging and expanding (Rezaeipanah et al. 2021; Rezaeipanah, Nazari, and Ahmadi 2019). This computing is provided as a tool to respond to the needs of users and users can use its services on the internet without spatial dependence. It is also used for places where dynamic resource provision and the use of virtualization technology are important (Ali, Affan, and Alam 2019). Virtualization is the main technology as a service solution. Virtualization is a way to get more performance out of a computer by sharing resources. Virtual machines (VMs) are an appropriate solution for full implementation of software-based virtualization. In fact, using VMs, a hardware with a suitable quality level is simulated and virtualized. This mechanism requires high speed internet and tasks in clusters. Cloud computing has a bright future but still has a problem that has not been resolved (Skarlat et al. 2016).

From a user's perspective, cloud computing is an abstract concept of extremely scalable and distant storage and computing resources. From a service provider perspective, cloud systems are based on a large set of computer resources and are allocated to applications on demand (Warneke and Kao 2011). Thus, cloud computing can be defined as a distributed template in which all resources are presented in dynamic scalability and virtualization as a service on the internet. The high scalability of cloud computing services allows users to increase and decrease resources at any time. Reducing costs is one of the most important benefits of a cloud computing service. Other advantages of cloud computing include high speed, security, reliability, and acceptable performance (Chien, Lai, and Chao 2019; Jula et al. 2021).

In cloud computing, Dynamic Resource Allocation (DSA) is the process of allocating resources available to applications over the internet (Elhoseny et al. 2018). DSA causes famine of services if allocation is not managed properly (Mahini et al. 2021). Resource provision solves the famine problem by allowing service providers to manage resources in each module. DSA strategies are for allocating appropriate resources in the cloud environment to meet the needs of applications. DSA strategies and input parameters to DSA are different based on services, infrastructure, and the nature of applications (Xu et al. 2019). Figure 1 shows a classification of DSA-based strategies for cloud computing.

Every request sent by users to service providers takes over part of their resources. The DSA problem is a major challenge in cloud environments and is directly related to energy consumption, service providers' profits

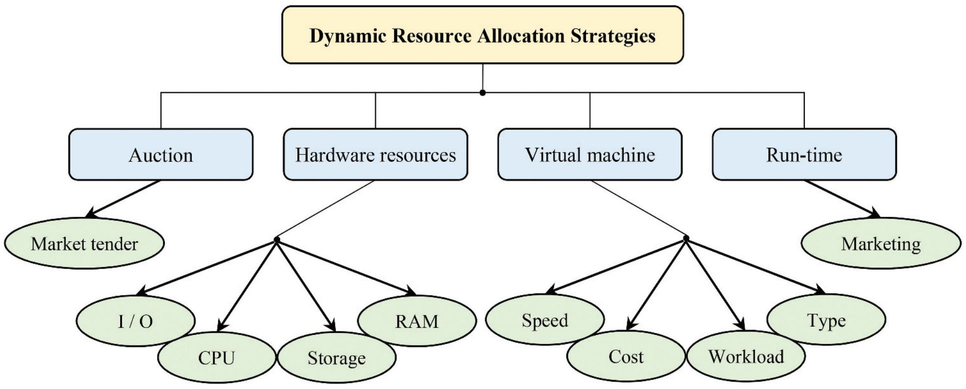

Figure 1. DSA-based strategies.

and users' costs (Taher et al. 2019). Hence, much task has been done in the field of DSA, reducing the number of resources utilization, load balancing and resource integration. Cloud customers need to receive quality, low cost and reliable services. Basically, service cost, Quality of Service (QoS), and service reliability depend on the DSA process in the cloud computing environment (Elhoseny et al. 2018; Taher et al. 2019). In cloud computing, multiple users can request a number of cloud services simultaneously. Hence, the need for algorithms to optimally manage DSA in the cloud with certain QoS requirements is essential. Cloud service users and providers have different goals and needs (Elhoseny et al. 2018). Users try to get the cloud services they need with QoS guaranteed and at the lowest cost. On the other hand, service providers also strive to have the highest return on investment. Therefore, a model with automated DSA capability in which the interests of both entities are considered can be very effective.

DSA in cloud computing is a mechanism that aims to meet the requirements of applications (Faraji Mehmandar, Jabbehdari, and Haj Seyyed Javadi 2020). In addition, the DSA management mechanism should examine the current state of each resource in the cloud environment in order to provide algorithms for optimal allocation of physical/virtual resources and reduction of operating costs (Xu et al. 2019). It is clear that due to the scale and complexity of these systems, centralized assignment of tasks to servers makes it impossible to consider specific solutions. Also, due to the increasing growth of data in data centers and the need to achieve the desired QoS, there is a need to provide solutions to increase the productivity of service providers. In the case of DSA, on the one hand, the needs of users should be considered and on the other hand, the available resources should be utilized to the maximum. As the number of cloud users increases, so do the resources that need to be allocated. Therefore, computing resources should be timed and allocated in such a way that both providers get the most out of the resources and users get the applications they need at the lowest cost (Naha et al. 2020; Pawar and Wagh 2012).

In this paper, solving DSA problem in cloud environment using Firefly Algorithm (FA) is presented as an optimization approach. Our goal is to improve this algorithm in order to optimize the workload balance for the initial population production. The proposed method is called IFA-DSA, and in addition to balancing the workload between existing VMs, it also reduces task completion time by selecting appropriate objectives. In order to achieve the goals, set for the production of the initial population in the FA, a heuristic method is used instead of a random approach. In IFA-DSA, the priority of each task is determined according to the pay as you use model and a fuzzy approach.

The main contribution of this paper is as follows:

- Development of FA to solve DSA problem as a multi-objective optimization problem
- Prioritize requests based on the pay as you use model and a fuzzy approach to improve load balance
- Evaluation of the proposed method with extensive simulation using Matlab software

The rest of the paper is organized as follows: Literature review on the DSA problem is compiled in Section 2. Section 3 provides background on problem formulation, objective functions, assumptions, and FA. An overview of the proposed method for solving the DSA problem in Section 4. Section 5 describes the simulation results and experimental results. Finally, Section 6 concludes this paper.

Literature Review

Currently, various studies have been conducted on the management of energy resources in cloud centers. Two types of energy saving algorithms (Khorsand et al. 2019; Panda and Jana 2019) and energy efficiency algorithms (Masdari et al. 2020; John 2020) are more important to solve the problem of high energy consumption in data centers. The main idea in these algorithms is to reduce energy consumption in data centers. For example, Ibrahim et al. (2020) proposes two heuristic algorithms to reduce energy consumption. These two algorithms are based on a simple local optimization method but do not consider the Service Level Agreement (SLA) violation. Sheikh and Pasha (2019) proposed an energy-saving resource allocation algorithm for cloud data centers. The results of this study show that the minimum or maximum runtime of tasks is not suitable for saving energy consumption. Therefore, this algorithm does not meet the needs of users in terms of energy consumption. The FCFS (First Come First Serve) method was proposed by Jaiganesh et al. (2015) to evaluate the performance of cloud services by optimizing task scheduling. In this method, the priority queuing method has been used to improve the scheduling system. Task Scheduling using a novel architecture with Dynamic Queues based on hybrid algorithm using Fuzzy Logic and Particle Swarm Optimization algorithm (TSDQ-FLPSO) was proposed by Ben Alla et al. (2016). TSDQ-FLPSO is a new architecture with dynamic queues based on fuzzy system and Particle Swarm Optimization (PSO) for scheduling tasks in cloud computing.

In recent years, many studies have been presented to solve the DSA problem in the haze environment, but more research is still needed. Because failure in DSA increases costs and delays Ibrahim et al. 2020 Most methods are based on heuristic and meta-heuristic approaches and use evolutionary algorithms (Skarlat et al. (2016); (Rajabion et al. 2019); (Liu et al. 2022). In this section, we will review the research conducted on the DSA problem.

Kumar and Kumar (2019) reviewed common load balancing algorithms in cloud computing. One of the most widely used of these algorithms is Min-Min. In this algorithm, first the minimum completion time for all tasks is found and then the minimum value is selected. This time is the minimum time between all tasks on all available resources. Finally, the task on the VM is scheduled according to the minimum time. However, Min-Min can lead to resource famine. Aghdashi and Mirtaheri (2019) presented a hybrid scheduling strategy for managing and processing medical data in cloud resources. The authors used a combination of Genetic Algorithm (GA) and PSO to load balancing and allocate resources fairly. In this strategy, cloud servers have the ability to process different user requests in parallel.

Elhoseny et al. (2018) presented a scheduling and load balancing model based on the concept of cloud segmentation. The authors use the round robin algorithm and game theory to select a service with a lower workload and higher execution speed to solve the DSA problem. Here, the round robin time slice is done according to the better server selection. Alelaiwi (2017) presented a scalable model based on Integer Linear Programming (ILP) to identify strategies and decisions in the cloud federation. In this model, user requests are sent at all levels of the cloud federation and they are easily guided. The profits from this model outsource resources to support other cloud federations. Mahini et al. (2021) proposed an approach to the allocation of multilayer resources. In this method, resource allocation to tasks is performed on a server of the cluster based on operating levels. This method improves efficiency by 18% and runtime by 10%.

Rajabion et al. (2019) proposed a resource allocation solution using cloud computing and the Internet of Things (IoT) for real-time and batch processing. Here, the IoT is responsible for real-time data processing, where it sends heavy processing to the cloud if needed. Simulations on Amazon Web Services show that this model can reduce response time to requests. Taher et al. (2019) proposed an autonomous approach based on fuzzy hierarchical technique to provide resources for multi-layered applications in the cloud. In the analysis phase of this approach, the support vector machine (SVM) technique and linear regression (LR) have been used to predict the number of user requests. This approach significantly reduces the cost and response time.

Jula et al. (2021) proposed an approach based on workload analysis to provide resource efficiency. This approach combines the imperialist competitive algorithm (ICA) and fuzzy c-means (FCM) clustering to categorize user requests. The authors use the decision tree algorithm to efficiently allocate resources based on both productivity and workload criteria. This approach focuses only on the mapping and resource allocation phase, and hardware or software failure can lead to increased runtime. Alboaneen et al. (2020) proposed a meta-heuristic approach to Joint Task Scheduling and VM Placement (JTSVMP) in cloud data centers. JTSVMP consists of two parts: task scheduling and VM placement. The authors formulate and solve the problem in an optimized way using meta-heuristic algorithms. The proposed optimization process is aimed at scheduling tasks in VMs that have the lowest execution cost in a limited time. The meta-heuristic algorithms used include glowworm swarm optimization (GSO), moth-flame glowworm swarm optimization (MFGSO), and GA.

Aburukba et al. (2020) proposed a resource scheduling algorithm for managing IoT services using ILP. The authors use an improved GA to minimize delay to take into account the dynamic nature of the fog environment. The performance of this algorithm is compared with round robin, waited-fair and priority-strict techniques. The results of this method show a delay rate between 21.9% to 46.6%, which is promising. Faraji Mehmandar, Jabbehdari, and Haj Seyyed Javadi (2020) introduced a distributed computing framework for resource management in fog computing. Here, the MAPE-k control loop is responsible for providing IoT services. In this regard, the reinforcement learning technique for the decision-making module and SVM for the analysis module have been used. The results of this method show better average cost and latency compared to similar algorithms. However, this approach does not take into account the different QoS needs of the fog service.

Skarlat et al. (2016) provided a conceptual framework for providing resources in fog computing. The purpose of this framework is to optimize the scheduling of IoT services by considering delays in the allocation of computing resources, where applications and resources heterogeneity are taken into account. To solve this problem, a GA-based meta-heuristic method is proposed that reduces communication delays and makes better utilization of fog resources. Wang, Liu, and Jolfaei (2020) proposed a DSA approach for

Table 1. Summary of related works.

Reference	Objective	Algorithm used
Skarlat et al. (2016)	A conceptual framework for providing resources in fog computing	A meta-heuristic method based on genetic algorithm
Alelaiwi (2017)	A scalable model based on linear programming in the cloud federation	A multi-objective optimization model based on genetic algorithm
Elhoseny et al. (2018)	A scheduling and load balancing model based on the concept of cloud segmentation	Round robin and game theory
Kumar and Kumar (2019)	An overview of common load balancing algorithms in cloud computing	One of the load balancing algorithms is Min-Min.
Aghdashi and Mirtaheri (2019)	A hybrid scheduling strategy for managing and processing medical data in the cloud	Combining genetic algorithms and particle swarm optimization
Rajabion et al. (2019)	A DSA approach in cloud computing and the Internet of Things for real-time and batch processing	Analysis of different algorithms
Taher et al. (2019)	An autonomous approach based on fuzzy hierarchical technique for DSA in the cloud	Support vector machines and linear regression
Alboaneen et al. (2020)	Joint task scheduling and VM placement	Glowworm swarm optimization, moth-flame glowworm swarm optimization, and genetic algorithm
Aburukba et al. (2020)	A resource scheduling algorithm for managing IoT services using integer linear programming	Improved genetic algorithm
Faraji Mehmandar, Jabbehdari, and Haj Seyyed Javadi (2020)	A distributed computing framework for resource management in fog computing	Reinforcement learning and support vector regression techniques
Wang, Liu, and Jolfaei (2020)	A multi-objective optimization approach for DSA on cloud-based sensor networks	Chaotic firefly algorithm
Mahini et al. (2021)	A scheduling algorithm for multilayer DSA in the cloud	A formal definition of cloud federation integrity based on clustering
Jula et al. (2021)	An approach based on workload analysis to ensure resource efficiency	Imperialist competitive algorithm and fuzzy c-means clustering

sensor networks using the improved chaotic firefly algorithm (ICFA) in the cloud. This approach creates a multi-objective optimization model based on the interference analysis of the working scenario of the cognitive radio. Since the multi-objective model is a nonlinear convex optimization problem, this paper uses ICFA to solve it. ICFA can effectively achieve the optimal solution while reducing the complexity of the problem.

Table 1 summarizes the literature reviewed in terms of objectives and algorithms used. Although there are numerous works in the literature, some methods do not take into account the characteristics of the cloud environment and need to improve performance in several respects, including DSA.

Background

This section consists of four subsections. The first subsection is devoted to formulating the DSA problem. The second subsection reviews the objective functions in DSA. The third subsection refers to the system hypotheses and finally the fourth subsection deals with the problem-solving approach, i.e., FA.

Problem Formulation

In the pay as you use model, users pay more to request their task done in less time (Ponraj 2019). The processing units in cloud computing are VMs. In cloud computing, there are a number of physical servers that each contain multiple VMs. VMs as a computing unit of the cloud environment have different types such as computers, servers and networks so that each VM has multiple resources (Wang, Liu, and Jolfaei 2020). Let the resources in each VM be static and LAN-based communication where this allows parallel processing. Task scheduling aims to allocate appropriate VMs to user requests to maximize resource utilization. This is referred to as the DSA problem (Kumar, Kikla, and Navya 2022).

The cloud environment has a scheduling system for the proper distribution of tasks between physical servers (Elhoseny et al. 2018). Each physical server also has an independent scheduling system that divides tasks between VMs. Since each VM has several independent sources with parallel processing capability, several tasks can be assigned to one VM simultaneously (Kumar, Kikla, and Navya 2022). All tasks send by users are assumed to be computational and independent of each other. The most important goal of the scheduling system in the cloud environment is the load balance of each VM. Load balancing is the concept of ensuring the proper distribution of tasks in terms of computational volume between VMs (Wang, Liu, and Jolfaei 2020). The main purpose of scheduling algorithms in the cloud environment is to maximize resource efficiency and reduce tasks spanning time according to user requests (Liu et al. 2022). Therefore, the DSA problem is known as a multi-objective optimization problem in the cloud environment. Solving the DSA problem means finding the proper sequence of tasks to run on VMs so that it reduces the makespan and load balances the VMs (Kumar, Kikla, and Navya 2022).

In recent years, the issue of migration in the cloud environment has become one of the major challenges in scheduling systems (Rezaeipanah, Nazari, and Ahmadi 2019). When too much task is assigned to one VM, it becomes an Overloaded VM (OVM). Therefore, some existing tasks must be transferred from OVM to Lowloaded VM (LVM) (Ibrahim et al. 2020; Zhou, Hu, and Li 2016). This technique is known as migration, which results in a workload balance between VMs. Given that migration has a computational overhead, therefore reducing migration rates in the cloud environment and load balancing will improve scheduling (Zhou, Hu, and Li 2016). In cloud environment scheduling systems for DSA on tasks offered by users, the K physical server is available as $PS = \{p_1, p_2, \ldots, p_i, \ldots, p_K\}$. Each p_i consists of m_i VM as $VM_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,j}, \ldots, v_{i,m_i}\}$. Thus, VM_i represents a list of VMs on the physical server p_i. Also, $v_{i,j}$ represents the j th VM in the i th physical server. In general, the purpose of the scheduling system is to apply DSA to N independent tasks as $T = \{t_1, t_2, \ldots, t_N\}$ on M VMs. Here, M is the total number of VMs associated with the K physical server, where $VM = \{v_1, v_2, \ldots, v_M\}$.

Objective Functions

There are many factors in the DSA problem that are considered as objective functions by various researchers (Ponraj 2019). Some of these factors include total runtime, makespan, workload balance and migration rate (Zhou, Hu, and Li 2016). These factors are briefly discussed below. The completion time of a task depends on the runtime by the VM. Let $TT_{i,j}$ be the runtime of t_i on v_j, TT_j is the average total runtime of the tasks assigned to v_j, as defined in Eq. (1).

$$TT_j = \frac{1}{t_i \in v_j} \sum_{t_i \in v_j} TT_{i,j} \tag{1}$$

Where, $t_i \in v_j$ represents the number of tasks assigned to v_j. Accordingly, TT is the average of the total runtime for all virtual machines defined by Eq. (2).

$$TT = \frac{1}{M} \sum_{j=1}^{M} TT_j \tag{2}$$

Let CT_{ij} be the completion time of t_i on v_j and CT_{max} be the completion time of the last task (makespan). Here, makespan means the largest completion time between all tasks and is defined according to Eq. (3).

$$CT_{max} = \max\{CT_{ij} | i \in T,\ i = 1, 2, \ldots,\ N \ and \ j \in VM,\ j = 1, 2, \ldots, M\} \tag{3}$$

The workload of a VM is calculated according to the size of the tasks running on this VM in relation to the total size of tasks. In general, WL_j workload is related to v_j, which is determined based on the size of tasks in the v_j queue. Hence, WL represents the average workload of all VMs and is defined by Eq. (4).

$$WL = \frac{1}{M}\sum_{j=1}^{M} WL_j \tag{4}$$

In order to describe the scatter and load balance of different VMs, the load variance of VMs is defined as $\sigma(WL)$. The value of $\sigma(WL)$ is used to check the load balance of a cloud environment scheduling system. The lower the load variance of VMs, the better the distribution of tasks and therefore the better the load balance. Eq. (5) is defined for calculating $\sigma(WL)$.

$$\sigma(WL) = \sqrt{\frac{1}{M-1}\sum_{j=1}^{M}\left(WL_j - WL\right)^2} \tag{5}$$

The migration technique can be used when a VM has a large load (Rezaeipanah, Nazari, and Ahmadi 2019). In this technique, a number of tasks are removed from the OVM queue and transferred to the LVM. Increasing the migration rate increases the time it takes to complete a task. Therefore, a parameter called migration rate is introduced here and the cloud environment scheduling system seeks to minimize it. Migration is the ratio of the number of tasks transferred to the total number of tasks over a time period. If TS is the entire scheduling period in the cloud environment being tested, then the time periods can be considered as Eq. (6).

$$TS = [(t_0 - t_1), (t_1 - t_2), \ldots, (t_{k-1} - t_k), \ldots] \tag{6}$$

Here, $(t_{k-1} - t_k)$ refers to the k th time period. Therefore, the migration rate can be calculated based on the time period k. $\rho(k)$ is the migration rate over time k and is calculated as Eq. (7).

$$\rho(k) = \frac{n_k}{N}, k \in TS \tag{7}$$

Where, n_k indicates the number of tasks that have been transferred to other VMs over time period k.

System Hypotheses

The simulation start time in the scheduling system is considered to be zero seconds. This time is not taken into account due to the slight delay in preparing VMs. Therefore, the runtime of a task depends on the runtime the task on the VM. All tasks and resources are computational and independent. Here, each request from users includes details such as request send time per second, task size based on million instructions per second (MIPS), priority execution of request based on pay as you use. There is no time deadline for processing requests. Also, cloud environment specifications such as number of physical servers, number of active VMs and processing power of a VM in MIPS are available. All VMs are capable of performing more than one task (depending on available resources). We assume that the details of the user requests as well as the specifications of the VMs are the input to the system.

Firefly Algorithm

Optimization algorithms seek to find a solution in the search space that has the minimum (or maximum) value of the objective function (Hajipour, Khormuji, and Rostami 2016). In an optimization problem, the types of mathematical relationships between objectives, constraints, and decision variables characterize the difficulty of the problem. Objective optimization is often used by researchers to solve real-world problems (Yang and Deb 2009). In general, a better solution is achieved by setting several objectives in solving the problem. Multi-objective optimization is a process for solving a problem with simultaneous optimization of two or more objectives subject to constraints. In recent years, nature-inspired algorithms and biological processes have been the most powerful solutions to optimization problems (Hajipour, Khormuji, and Rostami 2016; Mirjalili and Lewis 2016; Yang and Deb 2009).

The FA was introduced by Yang (2009). The main idea of FA is inspired by the optical relationship between fireflies and their biological communication phenomenon. This algorithm can be considered as a manifestation of swarm intelligence in which the cooperation of low-intelligence members creates a higher level of intelligence (Rostami et al. 2021). FA is a meta-heuristic algorithm inspired by the behaviors of artificial fireflies and is formulated with three hypotheses: 1) fireflies can be attracted to each other regardless of gender, 2) the attractiveness factor is measured in proportion to the brightness of fireflies and can be increased or decreased based on the distance, and 3) when the brightness of both fireflies is the same, the fireflies move randomly (Shahidinejad et al. 2020). Fireflies in the FA work independently and are suitable for parallel processing. FA is one of the techniques recently used by researchers to solve optimization problems in dynamic environments (Yousif et al. 2022). Figure 2 shows the flowchart of the FA steps.

FA is one of the most efficient algorithms in solving hybrid optimization problems and is used in various fields of optimization problems. Today, FA has a variety of applications in various sciences (Yousif et al. 2022). Problems such as discrete optimization, anomaly, multi-objective and other issues are addressed by the behavior of fireflies (Yousif et al. 2022). Other applications of FA include the use of these algorithms in combination with optimization algorithms and other improved techniques. Evidence shows that most

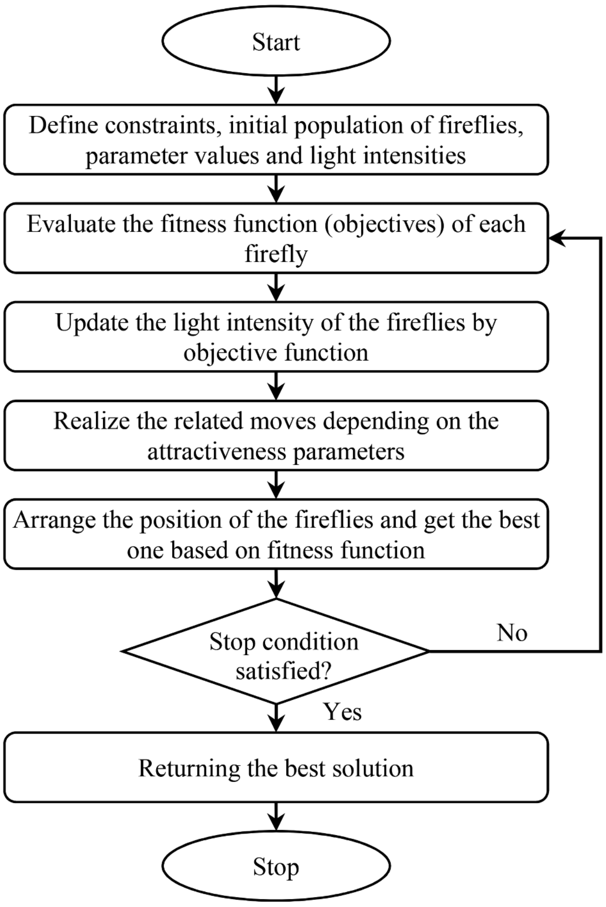
Start
Define constraints, initial population of fireflies, parameter values and light intensities
Evaluate the fitness function (objectives) of each firefly
Update the light intensity of the fireflies by objective function
Realize the related moves depending on the attractiveness parameters
Arrange the position of the fireflies and get the best one based on fitness function
Stop condition satisfied?
No
Yes
Returning the best solution
Stop

Figure 2. Flowchart of FA steps.

methods that have used the FA technique in combination with their method have performed better than other meta-heuristic algorithms (Wang, Liu, and Jolfaei 2020). In most cases, it uses a random search to reach a set of solutions. The FA focuses on generating solutions within a search space. Because this algorithm performs the search process randomly, it does not get stuck in local optimal points. Based on the advantages of this algorithm, we use FA as an optimization approach to solve the DSA problem in this study.

Proposed Method

In this paper, a combined FA-based optimization approach and fuzzy method are used to solve the DSA problem and improve task scheduling, which we call IFA-DSA. IFA-DSA tries to find the best sequence of tasks on VMs and ultimately improve DSA by considering different objectives and heuristic operators. Here, each firefly is a solution to the problem and specifies the

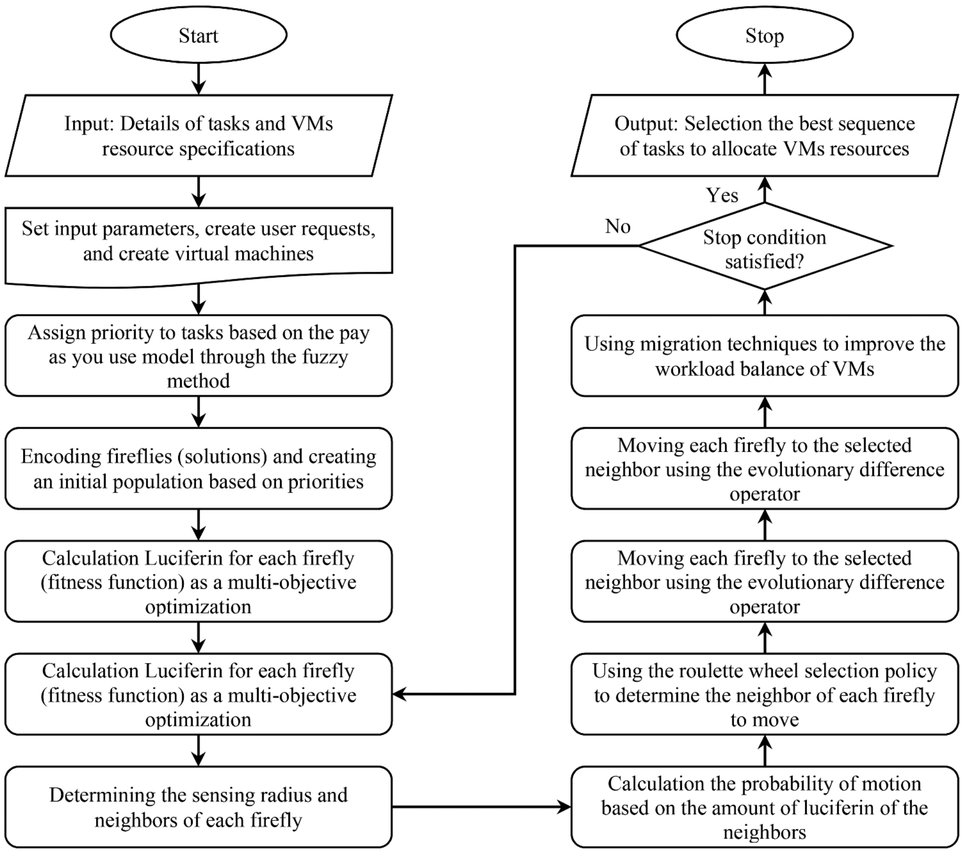
Start
Input: Details of tasks and VMs resource specifications
Set input parameters, create user requests, and create virtual machines
Assign priority to tasks based on the pay as you use model through the fuzzy method
Encoding fireflies (solutions) and creating an initial population based on priorities
Calculation Luciferin for each firefly (fitness function) as a multi-objective optimization
Calculation Luciferin for each firefly (fitness function) as a multi-objective optimization
Determining the sensing radius and neighbors of each firefly
Calculation the probability of motion based on the amount of luciferin of the neighbors
Using the roulette wheel selection policy to determine the neighbor of each firefly to move
Moving each firefly to the selected neighbor using the evolutionary difference operator
Moving each firefly to the selected neighbor using the evolutionary difference operator
Using migration techniques to improve the workload balance of VMs
Stop condition satisfied?
No
Yes
Output: Selection the best sequence of tasks to allocate VMs resources
Stop

Figure 3. IFA-DSA flowchart.

sequence of tasks on VMs. This sequence allocates the appropriate VMs resources to each task. In order to improve the DSA process, the priority of each task is determined by the pay as you use model. We use tasks priority to create an initial population in the FA in a heuristic method, where the goal is to properly distribute the task on VMs by a fuzzy approach.

Objectives function for improving the DSA problem are calculated as the luciferin value (fitness function) for each firefly. We consider four objectives including workload balancing, minimizing the completion time of the last task (i.e., makespan), minimizing the average runtime of tasks and reducing the migration rate, and formulate the problem as multi-objective optimization. In the next step, the sensor radius to determine the neighborhood is calculated based on the amount of luciferin, where the probability of fireflies moving toward the neighbors is calculated. This probability is defined in terms of higher luciferin (attractiveness). Finally, the movement of each firefly to a neighbor is more likely to be done using the evolutionary difference

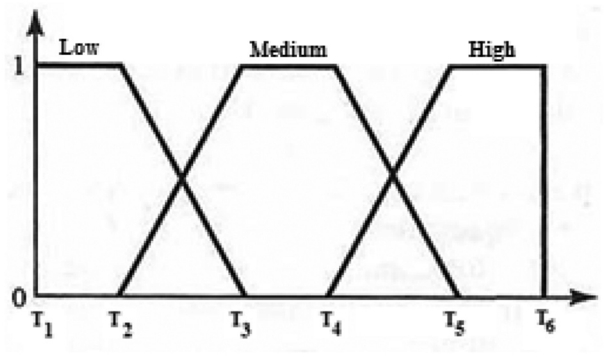

Figure 4. Trapezoidal membership function with three modes to prioritize tasks.

operator. In cases were increasing the workload on some VMs leads to workload imbalance, the load balance can be improved by the migration technique between VMs. The flowchart of IFA-DSA is shown in Figure 3.

Prioritize Tasks

This paper uses three modes, including Low, Medium and High, to prioritize user requests. These terms are fuzzy and we use fuzzy logic to assign them to user requests. Basically, the higher the user's payment and the smaller the computational size of the task, the higher the priority of the task to be performed. Let ω be the priority decision parameter, which is defined based on the ratio of the user's payment to the task size. We use a trapezoidal membership function with fixed threshold values (i.e., $T_1, T_2, \ldots, T_6$) to prioritize the tasks, as shown in Figure 4

The reason for estimating the input parameter is fuzzy in the form of Eq. (8) – Eq. (10). Here, the parameter ω determines the value of the fuzzy variables for the input value.

$$\mu_{Low}(\omega) = \begin{cases} 0 & if \;\; \omega \geq T_3 \\ \frac{T_3-\omega}{T_3-T_2} & if \;\; T_2 \leq \omega < T_3 \\ 1 & if \;\; \omega < T_2 \end{cases} \tag{8}$$

$$\mu_{Medium}(\omega) = \begin{cases} 0 & if \;\; \omega < T_2 \\ \frac{\omega-T_4}{T_5-T_4} & if \;\; T_2 \leq \omega < T_3 \\ 1 & if \; T_3 \; \leq \omega < T_4 \\ \frac{T_5-\omega}{T_5-T_4} & if \; T_4 \; \leq \omega < T_5 \\ 0 & if \;\; \omega \geq T_5 \end{cases} \tag{9}$$

t_1	t_2	...	t_i	...	t_N
v_a	v_b	...	v_j	...	v_z

Figure 5. Encoding structure of fireflies.

$$\mu_{High}(\omega) = \begin{cases} 0 & if\ \omega < T_4 \\ \frac{\omega - T_4}{T_5 - T_4} & if\ T_4 \leq \omega < T_5 \\ 1 & if\ \omega \geq T_5 \end{cases} \tag{10}$$

In this paper, rules in the form of Eq. (11) are used for the knowledge base.

$$R_k : If\ (x_1 is A_1)\ and. . .and(x_L is A_L) Then Y is G \tag{11}$$

Where, R_k refers to the rule k. x_i and A_i refer to the input parameter i and the assigned fuzzy set, respectively. Y is the output of the rule and G is a linguistic term of the fuzzy set belonging to it. Finally, L refers to the number of input parameters.

We use Mamdani fuzzy inference system to infer the winning rule. The degree of compatibility of the input parameters $X = \{x_1, x_2, \ldots, x_L\}$ with the premise of rule k is calculated based on Eq. (12).

$$\mu_k(X) = \prod_{i=1}^{L} \mu_{A_i(x_i)} \tag{12}$$

Where, L is the number of parameters and A_i is the fuzzy set i. In fact, A_i contains $\{A_{low}, A_{middle}, A_{high}\}$.

According to this method, the winning rule is selected based on the degree of confidence factor for decision making. Based on the fuzzy system, it is assumed that t_h, t_m, t_l are the tasks requested by users with high, medium and low priorities, respectively, which should be assigned to appropriate resources.

Encoding Fireflies and Initial Population

Each firefly is equivalent to a solution in the search space. The DSA problem encoding for each firefly is defined as a vector of length N, where N represents the total number of tasks. The content of each element of the vector is a VM assigned to the corresponding task. Figure 5 shows the representation structure of fireflies in the DSA problem. According to this definition, the i th element of this vector represents the j th VM assigned to the i th task.

The FA begins its work by producing an initial population of solutions. In classical FA, the initial population is created randomly, but in the case of DSA, this leads to a poor distribution of tasks between VMs. To solve this problem, we create the initial population on a heuristic basis based on the priority of tasks. Here the VM assigned to each task is determined based on the workload. VM selection for all three types of tasks with t_h, t_m, and t_l priorities are determined by Eq. (13) – Eq. (15).

$$t_h \rightarrow v_d | min\left(\sum\nolimits^{t_h}\right) \in VM \tag{13}$$

$$t_m \rightarrow v_d | min\left(\sum\nolimits^{t_h} + \sum\nolimits^{t_m}\right) \in VM \tag{14}$$

$$t_l \rightarrow v_d | min\left(\sum\nolimits^{T}\right) \in VM \tag{15}$$

Where, v_d is the VM assigned to the task, where $d = 1, 2, \ldots, M$.

Algorithm 1. Proposed heuristic method for creating an initial population

```
Input: $N_P$: Population size; $M$: Number of VMs; $N$: Number of tasks.
Output: $Pop(N_P, N)$ as the initial population.
1:  $Pop(N_P, N) \leftarrow$ null.
2:  for $i = 1$ to $N_P$ do
3:      for $j = 1$ to $N$ do
4:          if Task($t_j$) is high priority then
5:              $t_h \rightarrow v_d | min(\sum^{t_h}) \in VM$
6:              $Pop(i, j) = v_d$
7:          elseif Task($t_j$) is middle priority then
8:              $t_m \rightarrow v_d | min(\sum^{t_h} + \sum^{t_m}) \in VM$
9:              $Pop(i, j) = v_d$
10:         else
11:             $t_l \rightarrow v_d | min(\sum^{T}) \in VM$
12:             $Pop(i, j) = v_d$
13:         end
14:     end
15: end
16: The initial population created is $Pop$, as output.
```

When assigning a high-priority task (i.e., t_h), this task should be assigned to an LVM. This strategy ensures that a high-priority task selects a VM with the lowest number of high-priority tasks for processing. Medium priority tasks (t_m) should also select a VM with the lowest number of high and medium priority tasks. For low-priority tasks (t_m), selecting the VM with the least workload is emphasized. Algorithm 1 shows the pseudocode of the proposed heuristic method for creating the initial population. Here, N_P is the size of the firefly population.

According to Algorithm 1, line 1 defines the empty initial population in the *Pop* variable. Line 2 The process of creating solutions in the initial population is repeated as much as N_P. Lines 3 to 14 are related to creating a solution based on N existing tasks. In line 4, the priority of the tasks is checked in terms of high. High-priority tasks are assigned to the appropriate VM according to line 5. The selected VM is recorded in line 6 in the current solution. Similarly, lines 7 to 9 indicate the VM for medium-priority tasks, and lines 10 to 12 indicate this for low-priority tasks. Finally, the initial population in line 16 is returned as output.

Objective Function (Luciferin)

Fireflies have a radiance that attracts other fireflies to themselves. This radiance is known as luciferin. More luciferin for a firefly indicates its better position in the search space. In this paper, the luciferin of each firefly is formulated as a multi-objective function. Here, workload balancing, completion time of the last task (i.e., makespan), the average runtime of tasks and reducing the migration rate are considered as objectives function of the DSA problem. ℓ_i the value of luciferin is related to the i th firefly, which is calculated by Eq. (16).

$$\ell_i = a_1.TT + a_2.CT_{max} + a_3.\sigma(WL) + a_4.\rho(k) \tag{16}$$

Where, TT is the average runtime of tasks, CT_{max} is the completion time of the last task, $\sigma(WL)$ is the standard deviation of the workload balance and $\rho(k)$ is the migration rate. Also, a_1 to a_4 are the weights of these objectives, respectively.

Luciferin is released for each firefly during movement. This update reduces the radiance of fireflies due to time/iteration. Let $\ell_i(t)$ be the luciferin correspond to the i th firefly in the t th iteration and let $\ell_i(t-1)$ be the value for the $t-1$ iteration. Accordingly, Eq. (17) shows the firefly update process.

$$\ell_i(t) = (1-\rho)\ell_i(t-1) + \gamma\ell_i(t) \tag{17}$$

Where, ρ is a factor to model the gradual decline of luciferin and γ is defined to model the effect of fitness on luciferin.

Evolution of Fireflies

The process of population evolution in successive iterations is discussed in this section. Each firefly can have a number of neighbors based on its sensor radius. Based on these neighbors, each firefly selects a possible neighbor to move. The probability of selection is calculated based on higher luciferin and roulette wheel policy. In the DSA problem, which is a discrete problem, the amount of luciferin is used to determine the sensor radius and neighbors. In this paper, the sensing radius (rs) of the i th firefly includes all fireflies whose luciferin levels are in the range $\ell_i(t) \mp \theta$. Here, θ is the radius of decision of fireflies.

Let E_i be the set of i th firefly neighbors. As shown in Eq. (18), p_{ij} is the probability of the i th firefly moving toward the j th firefly, where $j \in E_i$. After calculating the probabilities for all neighbors, a neighbor is selected based on the roulette wheel policy (Rezaeipanah, Nazari, and Ahmadi 2019).

$$p_{ij} = \frac{\ell_j(t) - \ell_i(t)}{\sum_{k \varepsilon E_i} \ell_k(t) - \ell_i(t)}, j \in E_i \tag{18}$$

Let X_j be the firefly selected to move the X_i firefly. In this paper, the movement process is performed based on an evolutionary difference operator. The purpose of this operator is to create a new position vector for X_i based on X_j. X^{new} is a new position vector and is generated according to the difference between X_i and X_j. Here, $x_i^{new}(k)$ is the new position in the k th element of X_i and is calculated as Eq. (19).

$$x_i^{new}(k) = \begin{cases} x_i(k) & R < RND \\ x_i(k) + F \times (x_i - x_j) & Otherwise \end{cases} \tag{19}$$

Where, RND is a random number between [0–1] and R is a numerical constant smaller than one that regulates the rate of population convergence. F is a scale factor that controls the large differences between the values generated in the search space. F is calculated dynamically according to Eq. (20).

$$F = \frac{C_1 \times RND}{max(x_i(k), x_j(k))} \tag{20}$$

Where, C_1 is a constant less than one. The effect of this approach is to allow each member of the population to fluctuate within appropriate ranges to reach optimal solutions.

Migration Technique

When an increase in load on some VMs leads to an imbalance in the load of the system, the migration technique is used to improve the workload. However, increased migration leads to increased computational costs. Once an OVM is created, tasks must be moved from OVM to LVM. This means migration, and eventually results in a set of balanced VMs (BVMs). In this paper, whenever the load difference between LVM and OVM exceeds the ψ threshold, a task is randomly selected from OVM and transferred to LVM.

Table 2. Scenarios defined for the DSA problem.

Scenario	Number of tasks (N)	Number of VMs (M)
Scenario 1	100	5
Scenario 2	50	10
Scenario 3	500	60
Scenario 4	1000	100

Table 3. Configuring the cloud environment for the DSA problem.

Parameter	Value
Task size	Randomly between 500 and 2000
Power VMs	Randomly between 100 and 1000
Number of resources in VMs	Randomly between 1 and 4

Stop Condition

After an iteration of the FA, the current population has evolved over the old population. This process is repeated until a stop condition is reached. In this paper, the stop condition of the algorithm is defined based on the fixed number of iterations with the variable $Iter_{max}$.

Simulation Results

This section is related to the evaluation and comparison of the proposed IFA-DSA method in solving the DSA problem in the cloud environment. Evaluation and comparison are based on various criteria such as workload balancing and makespan. FCFS (Jaiganesh et al. 2015), TSDQ-FLPSO (Ben Alla et al. 2016) and ICFA (Wang, Liu, and Jolfaei 2020) methods were used to compare the performance of the IFA-DSA. In addition, the performance of the FA is evaluated compared to other evolutionary algorithms (i.e., GA and PSO). The simulation was performed by MATLAB R2019a on a Dell Inspiron laptop with an Intel Core i7-11370 H processor in 12MB cache, 4.8 GHz and 16GB memory. Various scenarios have been considered to evaluate the proposed method in the cloud environment, as shown in Table 2. For each scenario, the cloud environment configuration for the DSA problem is based on Table 3.

The IFA-DSA algorithm has different input parameters. Here, the priority decision threshold values are set as equal divisions between 0 and 1. The values of other parameters based on Ben Ben Alla et al. (2016) and Wang, Liu, and Jolfaei (2020) are as follows: $\psi = 0.1$, $C_1 = 0.8$, $R = 0.1$, $\theta = \mp 0.5$, $\gamma = 0.9$, $\rho = 0.15$, $a_1 = 0.5$, $a_2 = 0.1$, $a_3 = 0.1$, $a_4 = 0.3$, $N_P = 40$, $Iter_{max} = 200$.

The proposed method creates the initial population in a heuristic method based on a fuzzy approach and taking into account the priority of tasks. This is done with the aim of distributing the workload at the beginning of the scheduling process. Figure 6 shows the workload created on each VM for

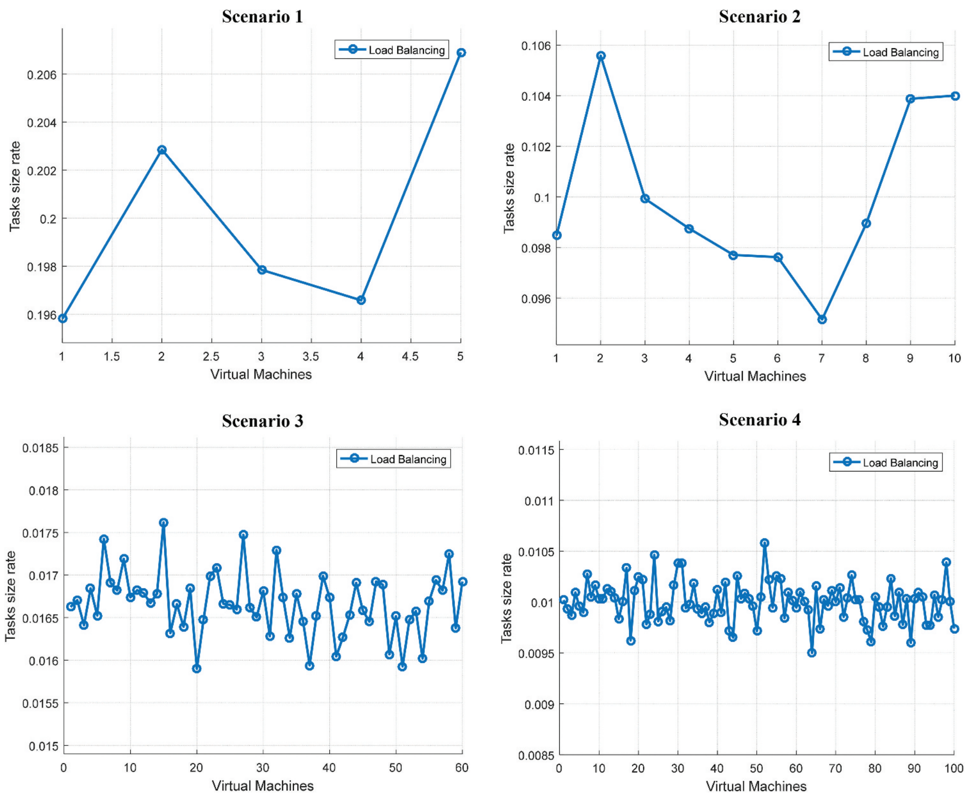

Figure 6. VMs load balancing at the beginning of the scheduling process.

the defined scenarios. Here, the workload is calculated based on the tasks size assigned to a VM relative to the total workload. Also, the workload of each VM is reported on an average basis for the entire population of fireflies. The results of this simulation show that the workload in all VMs is almost at the same level. Hence, the heuristic approach provides the apt load balancing at the beginning of the scheduling process.

The tasks distribution process on VMs is based on the priority of each task. the goal is to have almost the same number of tasks assigned to each VM with different priorities. In general, the IFA-DSA allocates sequence tasks to VMs in creating the initial population in such a way that the number of tasks with different priorities on VMs is fair. The results of the study of this problem are shown in Figure 7 for different scenarios. As illustrated, it is fair to distribute tasks with different priorities between VMs in all scenarios. For example, in Scenario 2, the average number of tasks assigned with low priority is approximately 11.7, and this value is 11.3 and 12.0 for medium and high priority, respectively. Based on these results, the proximity of the number of tasks with different priorities indicates the proper distribution of tasks among VMs by IFA-DSA.

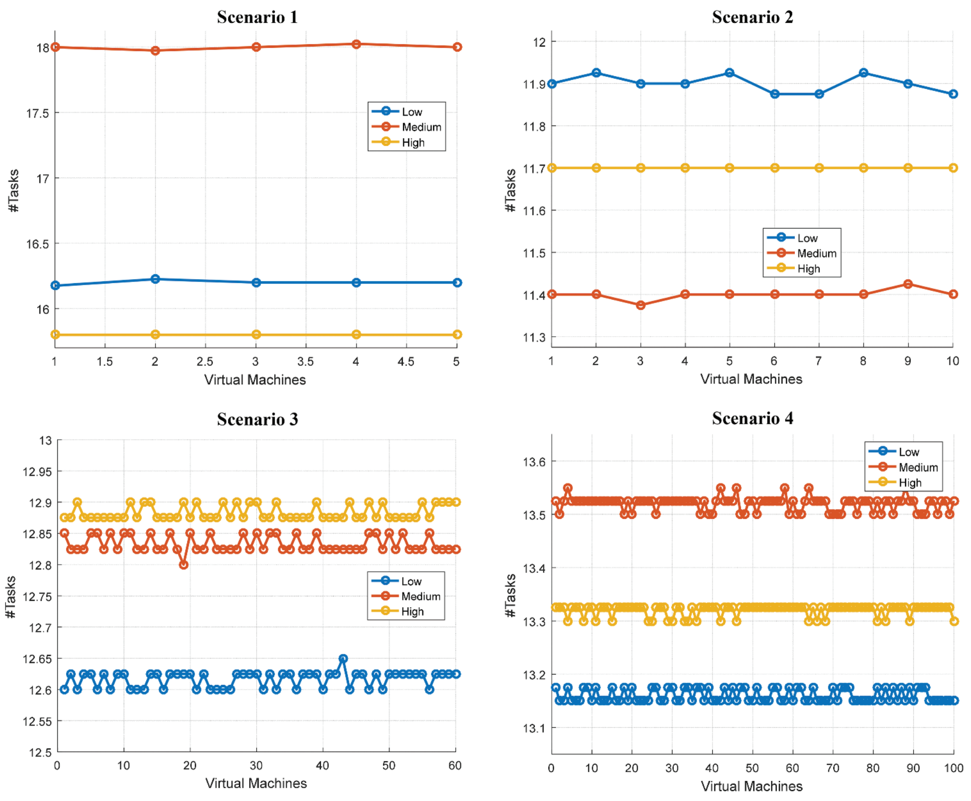

Figure 7. Distribution of tasks with different priorities on VMs.

The FA performance is then evaluated in comparison with GA and PSO. This evaluation is based on the makespan metric for the various scenarios shown in Figure 8. Here, the comparison based on the convergence rate of different methods in reducing makespan is emphasized. The results of this comparison show the superior performance of FA compared to GA and PSO during the optimization process. The reason for this advantage is the use of a heuristic method to generate the initial population and the proper distribution of tasks among VMs. The proposed method has a population of best quality at the start of optimization, and load balancing on VMs accelerates convergence during evolution.

Another experiment evaluated the performance of IFA-DSA compared to similar methods (i.e., FCFS, TSDQ-FLPSO and ICFA). These results are reported based on the makespan criteria and with a different number of tasks (i.e., 100, 200, 300, 400 and 500). In this comparison, the number of VMs is fixed and set to 50. The results of this experiment are shown in Figure 9. As illustrated, the superiority of IFA-DSA with makespan is less obvious than other methods. The proposed method is superior to other methods with a number of different tasks except 500. Here, ICFA performs better than IFA-DSA when the number of tasks is 500. Hence, the IFA-DSA

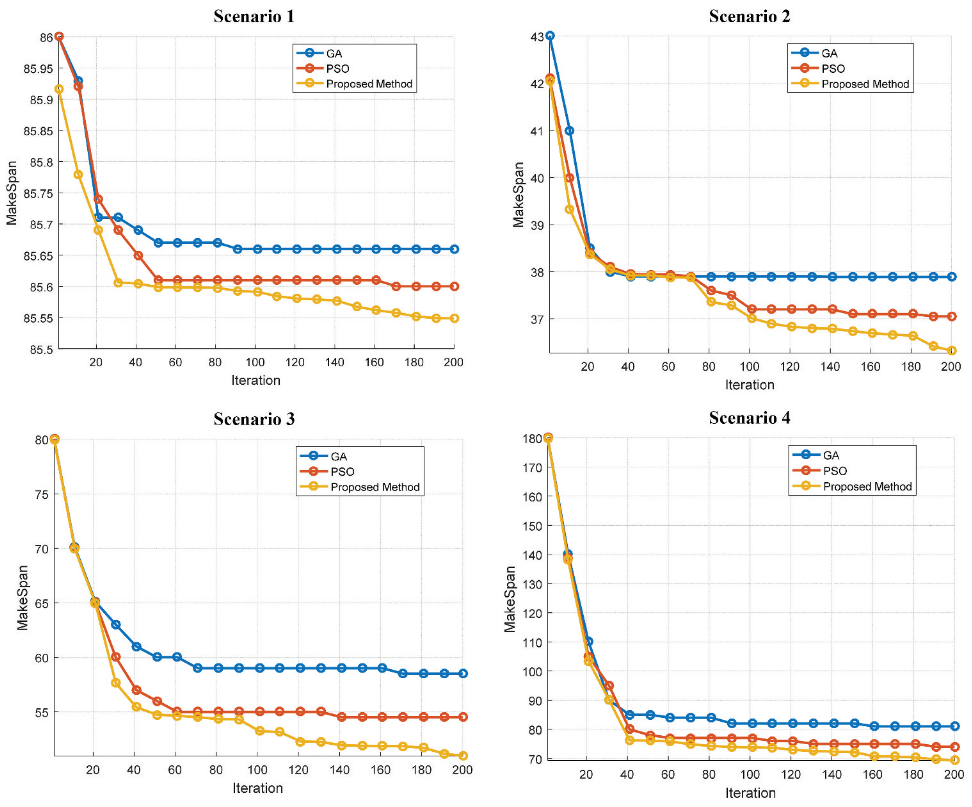

Figure 8. Comparison of FA with GA and PSO in the proposed method.

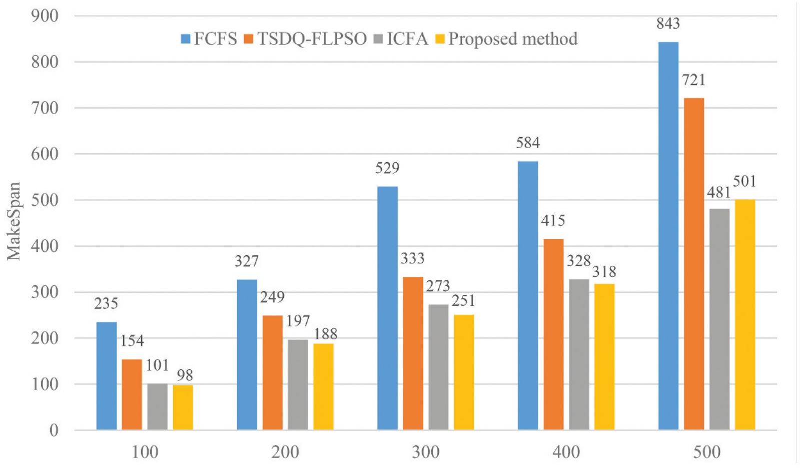

Figure 9. Comparison of IFA-DSA with similar methods in the makespan criteria.

Table 4. Statistical analysis of the proposed method in comparison with other methods.

	FCFS				TSDQ-FLPSO				ICFA			
Evaluation criteria	Mean	SD	*t*-value	*p*-value	Mean	SD	*t*-value	*p*-value	Mean	SD	*t*-value	*p*-value
Workload balance	2.16	3.73	2.54	0.016	2.95	3.92	5.98	0.009	3.01	2.67	2.29	0.01489058
Makespan	50.16	5.43	2.73	0.018	53.04	5.30	2.64	0.015	53.45	5.23	2.32	0.0256673

Table 5. Parameters used by different methods for DSA.

Parameters	FCFS	TSDQ-FLPSO	ICFA	IFA-DSA
Workload balance	√	√	√	√
Being fair	-	√	-	-
Efficiency	√	-	-	-
Utilization of resources	-	-	√	-
Run-time	-	-	-	√
Energy consumption	-	-	√	-
Processing cost	√	-	-	√
Response time	√	√	√	-
Prioritize tasks	-	-	-	√

performs worse than the ICFA as the number of tasks increases, although this is negligible. Overall, the average makespan results show that IFA-DSA performs 95%, 39% and 3% better than FCFS, TSDQ-FLPSO and ICFA methods, respectively.

Table 4 shows the proposed method statistical analysis for DSA in the cloud using paired *t*-test, which determines the significance level of the proposed method compared to other methods. These results are reported based on an average of 4 scenarios. Evaluation criteria used in experiments such as load balancing and makespan are considered in this analysis. Statistical parameters include difference of mean and standard deviation (SD) between IFA-DSA and comparable methods (i.e., FCFS, TSDQ-FLPSO and ICFA) based on *t*-value and *p*-value. In this analysis, a significant level of $p < 0.05$ is considered for paired *t*-test. Therefore, when the *p*-value is less than 0.05, it can be concluded that the difference between the proposed method and a similar method is significant for an evaluation criterion. The results show that there is a significant difference between the proposed method and other methods, because for all evaluation criteria, the *p*-value is less than 0.05.

In a DSA system in a cloud environment, various quality parameters such as runtime, cost, load balance, energy consumption, fairness, resource efficiency, etc. must be considered. However, it is not possible to consider all of these parameters in practice. In a parametric comparison, the position of the proposed IFA-DSA method in comparison with FCFS, TSDQ-FLPSO and ICFA methods is investigated. The results of this study are presented in

Table 5. According to all the tests performed, IFA-DSA performs better than other methods compared in most cases and has reported promising results in other cases.

Conclusion

Cloud computing is a model based on large computer networks that provides a new model for the supply, consumption and provision of information technology services using the internet. Cloud centers provide their resources with a technology called virtualization, which can provide dynamic flexibility in the DSA. In this regard, virtualization technology allocates server resources dynamically and based on user requests. This technology allows simultaneous access to cloud services for a large number of users. Therefore, it is essential to provide sufficient resources for each application. Given the limited resources available, it is crucial for cloud service providers to manage resource allocation to consumers, which is dynamically changing. In DSA, various factors such as QoS, price, fairness, profit and workload balance are emphasized. The process of allocating resources is a complex one, as both the consumer and the cloud server seek to make the most profit. Inspired by evolutionary algorithms, this paper presents a DSA-based approach to efficient load balancing in the cloud environment. The proposed method has the advantages of both FA and fuzzy approaches. FA has been efficiently improved to create the initial population by the fuzzy approach. The proposed method provides the DSA process with efficient load balancing and proper runtime. The simulation results confirm the effectiveness of the proposed method and show its superiority over similar methods. For future work, the proposed method can be configured to perform cloud task scheduling to achieve some objectives such as maximizing the resource utilization and reducing SLA violation. In addition, the proposed method can be applied to emerging real-world applications such as home health care systems.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

ORCID

Mostafa Ghobaei-Arani http://orcid.org/0000-0003-2639-0900
Musa Mojarad http://orcid.org/0000-0002-2218-2170

References

Aburukba, R. O., M. AliKarrar, T. Landolsi, and K. El-Fakih. 2020. Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing. *Future Generation Computer Systems* 111:539–51. doi:10.1016/j.future.2019.09.039.

Aghdashi, A., and S. L. Mirtaheri. 2019, April. A survey on load balancing in cloud systems for big data applications. In I*nternational congress on high-performance computing and big data analysis,* Lucio Grandinetti, Seyedeh Leili Mirtaheri, Reza Shahbazian editors, 156–73. Cham: Springer.

Alboaneen, D., H. Tianfield, Y. Zhang, and B. Pranggono. 2020. A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers. *Future Generation Computer Systems* 115:201–12. doi:10.1016/j.future.2020.08.036.

Alelaiwi, A. 2017. A collaborative resource management for big IoT data processing in Cloud. *Cluster Computing* 20 (2):1791–99. doi:10.1007/s10586-017-0839-y.

Ali, S. A., M. Affan, and M. Alam (2019). A study of efficient energy management techniques for cloud computing environment,9th *International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp.13–18), IEEE, Noida, India.

Ben Alla, H., S. Ben Alla, A. Ezzati, and A. Mouhsen. 2016, May. A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing. In *International symposium on ubiquitous networking,* Rachid El-Azouzi, Daniel Sadoc Menasche, Essaïd Sabir, Francesco De Pellegrini, Mustapha Benjillali editots,205–17. Singapore: Springer.

Berahmand, K., E. Nasiri, Y. Li, and Y. Li. 2021. Spectral clustering on protein-protein interaction networks via constructing affinity matrix using attributed graph embedding. *Computers in Biology and Medicine* 138:104933. doi:10.1016/j.compbiomed.2021.104933.

Chien, W. C., C. F. Lai, and H. C. Chao. 2019. Dynamic resource prediction and allocation in C-RAN with edge artificial intelligence. *IEEE Transactions on Industrial Informatics* 15 (7):4306–14. doi:10.1109/TII.2019.2913169.

Elhoseny, M., A. Abdelaziz, A. S. Salama, A. M. Riad, K. Muhammad, and A. K. Sangaiah. 2018. A hybrid model of internet of things and cloud computing to manage big data in health services applications. *Future Generation Computer Systems* 86:1383–94. doi:10.1016/j.future.2018.03.005.

Faraji Mehmandar, M., S. Jabbehdari, and H. Haj Seyyed Javadi. 2020. A dynamic fog service provisioning approach for IoT applications. *International Journal of Communication Systems* 33 (14):e4541. doi:10.1002/dac.4541.

Hajipour, H., H. B. Khormuji, and H. Rostami. 2016. ODMA: A novel swarm-evolutionary metaheuristic optimizer inspired by open-source development model and communities. *Soft Computing* 20 (2):727–47. doi:10.1007/s00500-014-1536-x.

Ibrahim, A., M. Noshy, H. A. Ali, and M. Badawy. 2020. PAPSO: A power-aware VM placement technique based on particle swarm optimization. *IEEE Access* 8:81747–64. doi:10.1109/ACCESS.2020.2990828.

Jaiganesh, M., B. Ramadoss, A. V. A. Kumar, and S. Mercy. 2015. performance evaluation of cloud services with profit optimization. *Procedia Computer Science* 54:24–30. doi:10.1016/j.procs.2015.06.003.

John, N. P. (2020, March). A review on dynamic consolidation of virtual machines for effective energy management and resource utilization in data centres of cloud computing. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, India, (pp. 614–19). IEEE.

Jula, A., E. A. Sundararajan, Z. Othman, and N. K. Naseri. 2021. Color revolution: a novel operator for imperialist competitive algorithm in solving cloud computing service composition problem. *Symmetry* 13 (2):177. doi:10.3390/sym13020177.

Khorsand R, Ghobaei-Arani M and Ramezanpour M. 2019. A self-learning fuzzy approach for proactive resource provisioning in cloud environment. *Softw: Pract Exper* 49(11): 1618–1642. doi:10.1002/spe.2737

Kumar, N., M. Kikla, and C. Navya. 2022. Dynamic resource allocation for virtual machines in cloud data center. In *Emerging research in computing, information, communication and applications*, N. R. Shetty, L. M. Patnaik, H. C. Nagaraj, Prasad N. Hamsavath, N. Nalini editors, 501–10. Singapore: Springer.

Kumar, P., and R. Kumar. 2019. Issues and challenges of load balancing techniques in cloud computing: A survey. *ACM Computing Surveys (CSUR)* 51 (6):1–35. doi:10.1145/3281010.

Liu, C., J. Wang, L. Zhou, and A. Rezaeipanah. 2022. Solving the multi-objective problem of iot service placement in fog computing using cuckoo search algorithm. In press. *Neural Processing Letters*. 1–32. https://doi.org/10.1007/s11063-021-10708-2

Mahini, A., R. Berangi, A. M. Rahmani, and H. Navidi. 2021. Bankruptcy approach to integrity aware resource management in a cloud federation. *Cluster Computing* 24 (4):3469–94. doi:10.1007/s10586-021-03336-x.

Masdari M, Gharehpasha S, Ghobaei-Arani M and Ghasemi V. 2020. Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. *Cluster Comput* 23 (4): 2533–2563. doi:10.1007/s10586-019-03026-9

Mirjalili, S., and A. Lewis. 2016. The whale optimization algorithm. *Advances in Engineering Software* 95:51–67. doi:10.1016/j.advengsoft.2016.01.008.

Naha, R. K., S. Garg, A. Chan, and S. K. Battula. 2020. Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment. *Future Generation Computer Systems* 104:131–41. doi:10.1016/j.future.2019.10.018.

Panda, S. K., and P. K. Jana. 2019. An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems. *Cluster Computing* 22 (2):509–27. doi:10.1007/s10586-018-2858-8.

Pawar, C. S., and R. B. Wagh (2012, December). Priority based dynamic resource allocation in cloud computing. In *2012 International Symposium on Cloud and Services Computing*, Mangalore, India, (pp. 1–6). IEEE.

Ponraj, A. 2019. Optimistic virtual machine placement in cloud data centers using queuing approach. *Future Generation Computer Systems* 93:338–44. doi:10.1016/j.future.2018.10.022.

Rajabion, L., A. A. Shaltooki, M. Taghikhah, A. Ghasemi, and A. Badfar. 2019. Healthcare big data processing mechanisms: The role of cloud computing. *International Journal of Information Management* 49:271–89. doi:10.1016/j.ijinfomgt.2019.05.017.

Rezaeipanah, A., P. Amiri, H. Nazari, M. Mojarad, and H. Parvin. 2021. An energy-aware hybrid approach for wireless sensor networks using re-clustering-based multi-hop routing. *Wireless Personal Communications* 120 (4):3293–314. doi:10.1007/s11277-021-08614-w.

Rezaeipanah, A., M. Mojarad, and A. Fakhari. 2022. Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic. *International Journal of Computers and Applications* 44 (2):139–47. doi:10.1080/1206212X.2019.1709288.

Rezaeipanah, A., H. Nazari, and G. Ahmadi. 2019. A hybrid approach for prolonging lifetime of wireless sensor networks using genetic algorithm and online clustering. *Journal of Computing Science and Engineering* 13 (4):163–74. doi:10.5626/JCSE.2019.13.4.163.

Rostami, M., K. Berahmand, E. Nasiri, and S. Forouzandeh. 2021. Review of swarm intelligence-based feature selection methods. *Engineering Applications of Artificial Intelligence* 100:104210. doi:10.1016/j.engappai.2021.104210.

Shahidinejad A, Ghobaei-Arani M and Esmaeili L. 2020. An elastic controller using Colored Petri Nets in cloud computing environment. *Cluster Comput* 23 (2): 1045–1071. doi:10.1007/s10586-019-02972-8

Sheikh, S. Z., and M. A. Pasha (2019, October). An improved model for system-level energy minimization on real-time systems. In *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Rennes, France, (pp. 276–82). IEEE.

Skarlat, O., S. Schulte, M. Borkowski, and P. Leitner (2016, November). Resource provisioning for IoT services in the fog. In *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*, Macau, China, (pp. 32–39). IEEE.

Taher, N. C., I. Mallat, N. Agoulmine, and N. El-Mawass (2019, April). An IoT-Cloud based solution for real-time and batch processing of big data: Application in healthcare. In *2019 3rd International Conference on Bio-engineering for Smart Technologies (BioSMART)*, Paris, France, (pp. 1–8). IEEE.

Wang, Z., D. Liu, and A. Jolfaei. 2020. Resource allocation solution for sensor networks using improved chaotic firefly algorithm in IoT environment. *Computer Communications* 156:91–100. doi:10.1016/j.comcom.2020.03.039.

Wang, X., X. Wang, H. Che, K. Li, M. Huang, and C. Gao. 2015. An intelligent economic approach for dynamic resource allocation in cloud services. *IEEE Transactions on Cloud Computing* 3 (3):275–89. doi:10.1109/TCC.2015.2415776.

Warneke, D., and O. Kao. 2011. Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. *IEEE Transactions on Parallel and Distributed Systems* 22 (6):985–97. doi:10.1109/TPDS.2011.65.

Xu, X., R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou. 2019. Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud. *IEEE Transactions on Industrial Informatics* 16 (9):6172–81. doi:10.1109/TII.2019.2959258.

Yang, X. S. 2009, October. Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*, Osamu Watanabe, Thomas Zeugmann editors,169–78. Berlin, Heidelberg: Springer.

Yang, X. S., and S. Deb (2009, December). Cuckoo search via Lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)*, Coimbatore, India, (pp. 210–14). IEEE.

Yousif, A., S. M. Alqhtani, M. B. Bashir, A. Ali, R. Hamza, A. Hassan, and T. M. Tawfeeg. 2022. Greedy firefly algorithm for optimizing job scheduling in iot grid computing. *Sensors* 22 (3):850. doi:10.3390/s22030850.

Zhou, Z., Z. Hu, and K. Li. 2016. Virtual machine placement algorithm for both energy-awareness and SLA violation reduction in cloud data centers. *Scientific Programming* 2016:1–12.