Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Signature GOA: A novel comfort zone parameter adjustment using fuzzy signature for task scheduling in cloud environment

Aboozar Zandvakili*[1], Najme Mansouri†[2] and Mohammad Masoud Javidi‡[3]

[1,2,3]Department of Computer Science, Shahid Bahonar University of kerman, Kerman, Iran.

ABSTRACT

Task scheduling in cloud computing plays an essential role for service provider to enhance its quality of service. Grasshopper Optimization Algorithm (GOA) is an evolutionary computation technique developed by emulating the swarming behavior of grasshoppers while searching for food. GOA is easy to implement but it cannot make full utilization of every iteration, and there is a risk of falling into the local optimal. This paper proposes a suitable approach for adjusting the comfort zone parameter based on the fuzzy signatures called signature GOA (SGOA) to balance exploration and exploitation. Then, we propose task scheduling based on SGOA by considering different objectives such as completion time, delay, and the load balancing on the machines. Finally,

*Keyword:*Task scheduling , Fuzzy signature , Multi-objective optimization.

AMS subject Classification:PACS 07.05.Kf and MSC 68

ARTICLE INFO

Article history:
Research Paper
Received 11, June 2020
Received in revised form 18, April 2021
Accepted 2 May 2021
Available online 01, June 2021

*zandvakili.a@gmail.com
†Corresponding author: N. Mansouri. Email: najme.mansouri@gmail.com
‡javidi@uk.ac.ir

1 Abstract continued

different algorithms such as Particle Swarm Optimization (PSO), Simulated Annealing (SA), Tabu Search (TS), and multi-objective genetic algorithm, are used for comparison. The results show that among all algorithms, SGOA can be successful in much less iteration.

2 Introduction

All graphs considered here are simple, finite, connected and undirected. For One way to provide various services through the Internet is cloud computing. Different services can be hardware or software. The hardware part includes storage facilities and servers. The software part includes databases and networks. The important thing is that all these services are performed remotely, which saves money, increases productivity, speed and efficiency, performance, and security. Due to these reasons, cloud computing is very popular.
But with all these advantages, task scheduling is a major challenge in cloud computing. Task scheduling is the technique of assigning some machines (VMs) to some tasks so that user requests are answered quickly. Because task scheduling does not have a solution of polynomial order, researchers have used metaheuristic algorithms such as PSO, SA, TS, and multi-objective genetic algorithm, to solve it. In the following, cloud computing and task scheduling briefly are described.

2.1 Cloud computing

Cloud computing is a new method of processing in which integrating a huge amount of computing resources in a virtualized and scalable way to create an integrated system and is delivered through internet communication networks. The focus of this model is to provide service to the user on-demand, without the user to need special equipment for processing or being aware of the location of this processing (24).
One of the advantages of cloud computing is that users can avoid the increasing costs and complexity of owning and maintaining technology infrastructure and instead pay for the feature they consume at the time of use. Figure 1, summarizes the advantages, disadvantages, and other parameters of cloud computing.
In Fig. 1, the service-based classification is presented as follows (38):
Infrastructure as a Service (IaaS): In this service, companies rent their required resources from servers. Users can run any program on leased servers without paying maintenance costs.
Platform as a service (PaaS): In this service, hardware and software tools are provided to users through the Internet. These tools are usually needed to develop applications. The hardware and software are hosted on the infrastructure of PaaS provider. Developers are released from installing internal hardware and software to develop or run new applications by PaaS.

Software as a service (SaaS): This service is a software distribution model in which applications are hosted by the provider.
In Fig. 1, the deployment models include the following sections:
Public cloud: Public cloud is a standard model of cloud computing service that is available to the general public. It is a collection of virtual resources. These resources are powered by hardware owned, managed, and organized by a third-party company. Resources in the public cloud are automatically provided and allocated to multiple clients through a self-service interface.
Private cloud: Private cloud is a special model of cloud computing that includes a separate, secure cloud-based environment that works for only one customer. In the private cloud model, the cloud (resource pool) is provided by only one provider organization with higher control that has privacy is available.
Hybrid cloud: Hybrid cloud considers a mix of private cloud and public cloud services. It can communicate between each separate service through proprietary software. Applying a hybrid cloud strategy will shift workloads due to fluctuations in needs and costs, and ultimately provide more flexibility and adaptability for jobs. Hybrid cloud is very popular since it allows an organization to manage its private data in the best possible way and keep important data on a private cloud and at the same time use the powerful computing resources of a managed public cloud. Unlike a multi-cloud approach that the manager controls each cloud system separately, a hybrid cloud relies on one level of management. Because cloud service providers are required to provide services in the shortest time to compete with each other, one of the most important challenges in cloud computing is the issue of task scheduling. Task scheduling is the decision of which task to do by which machine.

2.2 Task scheduling

An efficient task scheduling is a critical technique for resource management in a cloud environment with numerous users. The single objective task scheduling algorithm assumes only one criterion while multi-objective task scheduling considers two or more parameters (e.g., waiting time, load, and energy usage). The search space is in the form of Eq. (1).

$$SpaceSize = \frac{(M * N)!}{(M!)^N} \tag{1}$$

In Eq. (1), N: number of tasks and M: number of machines.
As a result, the task scheduling problem is one of the problems for which there is no solution with polynomial order. For this reason, solving it with precise methods is very time consuming and costly. Therefore, the best choice is to use meta-heuristic algorithms. In practice, the task scheduling problem can be modeled in different modes (Fig.2). In this classification, the classical type is divided into two sub-categories. In the classic type, the machines are the same. But in the flexible type, jobs can be divided into smaller sections (commonly known as tasks), and these sections can be executed on machines independently or in a specific order. In the flexible sub-category, all machines are capable

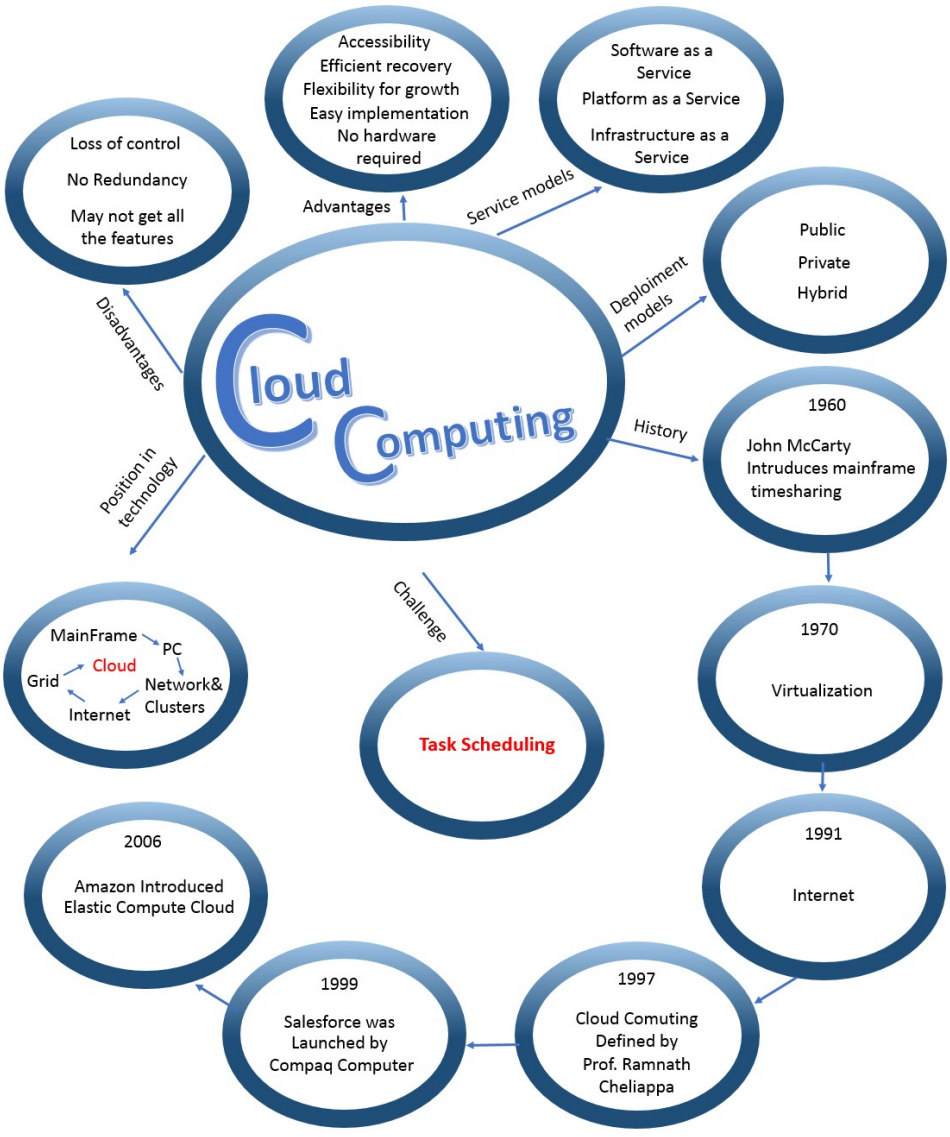

Figure 1: Cloud computing.

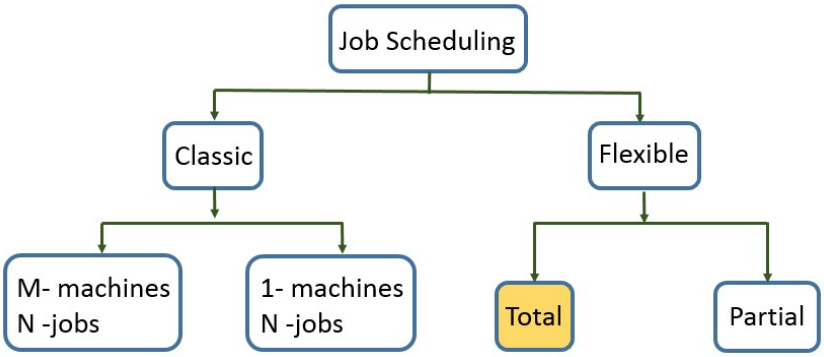

Figure 2: Task scheduling issue.

to do all tasks, but in the sub-category, some machines are not capable to do certain tasks. In this paper, machines are capable to perform all tasks, but each task is considered individually (each task does not include smaller parts).
To solve the problem in single-machine mode, different permutations of tasks can be considered and the best permutation can be selected. Several solutions have been proposed to solve these types of problems. 1) Accurate solution methods, 2) Heuristic methods. Although precise solutions is obtained by dynamic programming (20; 33) and branch and bound algorithms (4; 44), their implementation requires a lot of memory and time. In most recent works, prohibited search-based methods with specific neighborhoods have been proposed that have better results than exploratory methods (9). In conventional genetic algorithms, the combination operator is used as the basic operator to improve the performance of the algorithm and the progress of the algorithm depends a lot on this operator (54). In (28), using real data, the effect of ten combination operators on the mentioned problem has been investigated.
But the multi-machine mode is more complex. Undoubtedly, solving the task scheduling problem is very time consuming and has a high computational load due to the nature of these types of problems with precise methods, so an acceptable answer can be obtained by choosing the proper meta-heuristic algorithms. Because the proposed methods require a lot of computational time or complex mathematical calculations, using the Genetic Algorithm (GA) has been successfully started over the last 20 years (27; 58; 62; 66; 67).
For the task scheduling problem, different versions of optimization algorithms have been developed with superior performance. The authors in the research paper (2) used Cuckoo Search (CS) algorithm to reduce makespan, but this paper did not take QoS into account. In (35), the authors used an artificial Bee Colony (ABC) to minimize the completion time and total workloads of all devices. However, the response time or tardiness was not considered. In (64), PSO Algorithm is used for task scheduling problems. The authors optimized the execution cost, but load balancing and other effective QoS is not considered. In (56), ACO Algorithm is used for task scheduling problems. The proposed algorithm could reduce makespan and balance cloud clusters, but it still needs to consider some other QoS factors (e.g., deadline). The authors in the research paper (39) used GWO to

reduce makespan and energy usage, but this paper did not consider any constraints like the deadline, priority of applications, etc. In (70), a hybrid algorithm (SGA with CHC) is used for task scheduling problem. The presented work improved the completion time of tasks, but it did not pay attention to the effective parameters (e.g., energy, reliability, and cost). The focus of previous researches is to minimize the cost or completion time of the task scheduling without regarding the QoS metrics such as tardiness and the load balancing of the cloud servers. In addition, none of them considered the tardiness, the load balancing, and the completion time with each other. We tackle this problem of developing a scheduling algorithm in the cloud environment to optimize tardiness, completion time, and load balancing, using a meta-heuristic algorithm, named Signature Grasshopper Optimization Algorithm (SGOA) that is improved GOA.
In this paper, we select grasshoppers algorithm for solving task scheduling problem since:

- Effectively explore promising areas of a search space
- Have large-scale changes in the initial steps
- Tend to move locally in the final steps
- Gradually balance exploration and exploitation

The comfort zone parameter is an important factor in GOA, we optimize GOA using the fuzzy signature. Second, we use SGOA to solve the task scheduling problem in the cloud environment. Finally, SGOA compared with the SA, TS, PSO, GOA, and GA. The divisions of the used algorithms can be seen in Fig. 3 and Fig. 4.

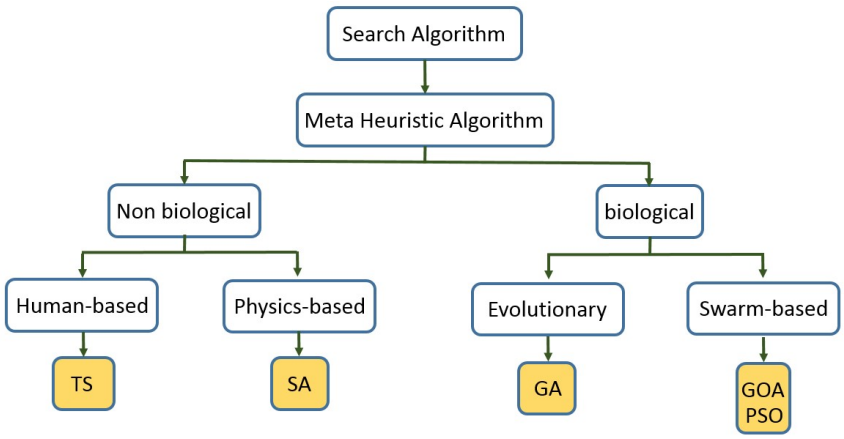

Figure 3: Classification of algorithms used according to the nature of the algorithm.

In this paper, the major contributions are as follows:

- Improve grasshopper optimization algorithm by the fuzzy signature and named SGOA.

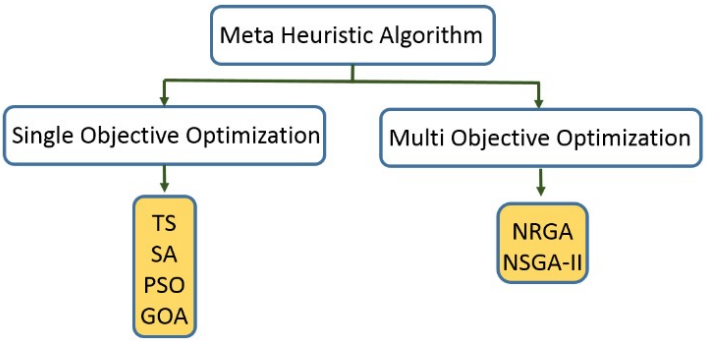

Figure 4: Classification of algorithms used in terms of the type of optimization.

- Design a multi-objective scheduling algorithm for finding an optimal mapping based on multiple conflicting objectives, namely makespan, load balancing, and tardiness.
- Use SGOA for scheduling in the cloud computing environment.
- The extensive experiments are performed to evaluate the proposed strategy with SA, TS, PSO, GOA, Non-dominated Ranking Genetic Algorithm (NRGA), and Non-dominated Sorting Genetic Algorithm II (NSGA-II).

The rest of the paper is organized as follows: Section 3 discusses the overview of the GOA, PSO - inertia weight, and Fuzzy Signature. In Section 4, the related works of the task scheduling approaches are explained briefly. Related definitions and task scheduling models are discussed in Section 5. The proposed SGOA is discussed in Section 6. In Section 7, the performance evaluation is provided. We conclude our work in Section 8.

3 Background knowledge

This section discusses the overview of GOA, PSO - inertia weight, and Fuzzy Signature.

3.1 Grasshopper Optimization Algorithm (GOA)

In the early articles, the word locust was used instead of grasshopper. Much research has been done on community, migration, velocity, population density, rolling structure, take-off zone, the zone of settlement, interior zone, and grasshoppers movement (16). The locust rolling movement is approximately 1 km long. Large clusters of locusts are often seen in cross-sections of 10 to100 km2 or more. The flight speed of the locust is usually in the range of 12 km/hr, and a maximum of 23 km/hr is visible. Locusts can move in the range of 5 to 50 km per day. In a typical group of locusts, there are approximately 107 to 109 locusts. The life cycle of grasshoppers can be seen in Fig. 5. Exploration and exploitation are very important in swarm-based algorithms. Usually in the initial steps, exploration is very important and in the final steps, exploitation is very important. In

(47) it is stated that these two phases are inherent in the behavior of grasshoppers. When the grasshoppers are immature, they have a smooth and continuous movement and have the role of exploitation. Instead, adult grasshoppers have random movements and play the role of exploration. The rolling grasshoppers swarm can be seen in Fig. 6

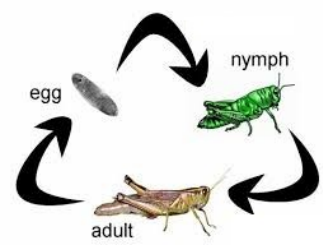

Figure 5: The life cycle of grasshoppers.

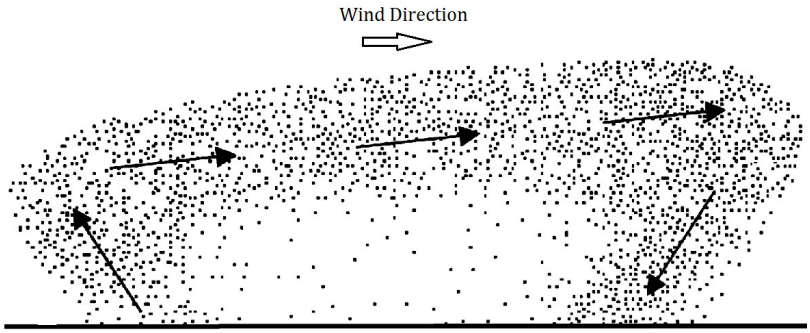

Figure 6: Schematic depiction of a rolling grasshoppers swarm (63).

The group behavior of grasshoppers was discussed in (63). After that, the mathematical equations of grasshoppers' motion were formulated in (60). Finally, in (47), the grasshopper optimization algorithm is presented. The mathematical model used to simulate the group behavior of grasshoppers is as follows (47):

$$X_i = S_i + G_i + A_i \tag{2}$$

In Eq. (2), X_i: the position of the $i-th$ grasshopper, S_i: the social interaction, G_i: the gravity force on the $i-th$ grasshopper, and A_i: the wind advection.

$$S_i = \sum_{j=1, i\neq j}^{N} s(d_{ij})\hat{d_{ij}} \tag{3}$$

In Eq. (3), d_{ij}: the distance between the $i-th$ and the $j-th$ grasshopper, s: the social forces, $\hat{d_{ij}}$: unit vector from the $i-th$ grasshopper to the $j-th$ grasshopper.

$$d_{ij} = \|x_j - x_i\| \tag{4}$$

$$\hat{d_{ij}} = \frac{x_j - x_i}{d_{ij}} \tag{5}$$

$$s(r) = fe^{\frac{-r}{l}} - e^{-r} \tag{6}$$

In Eq. (6), f : the intensity of attraction, and l: the attractive length scale. According to (47), we have chosen $l = 1.5$ and $f = 0.5$.
In Eq. (2), The G component is calculated as follows:

$$G_i = -g\hat{e_g} \tag{7}$$

In Eq. (7), g: the gravitational constant, and $\hat{e_g}$: a unity vector towards the center of the earth.
In Eq. (2), The A component is calculated as follows:

$$A_i = u\hat{e_w} \tag{8}$$

In Eq. (8), u: a constant drift, and $\hat{e_w}$: a unity vector in the direction of the wind.
Eq. (2) is not suitable for solving optimization problems in the same way. Because grasshoppers mostly reach the comfort zone quickly and do not gather at a specific point. An improved version of this equation is presented as ((9)) (47):

$${X_i}^d = c(\sum_{j=1,i\neq j}^{N} c\frac{ub_d - lb_d}{2} s(|x_j^d - x_i^d|)\frac{x_j - x_i}{d_{ij}}) + \hat{T}_d \tag{9}$$

In Eq. (9), the first c from the left is responsible for balancing the exploration and exploitation in the swarm and the second c decreases the attraction zone, comfort zone, and repulsion zone between grasshoppers. The ub_d and lb_d show the upper and lower bounds in dimension D, respectively. In Eq. (9), term $\hat{T}_d$ is used instead of terms G_i and A_i, which indicates moving towards the goal.

3.2 Particle Swarm Optimization (PSO)-inertia weight (W)

A population of simple members that interact locally with each other and with their environment is called a swarm behavior-based system. However, their local behavior towards each other creates public behavior, but there is no focused control over individual or group behavior. PSO is one of the most important algorithms in this field.
Kennedy and Eberhart introduced PSO based on the behavior of social animals such as fish and birds that live together in groups (29). In the PSO algorithm, the members of the population of answers are directly related to each other and reach the solution of the problem through the exchange of information. Using the following equations, the velocity and position of each particle can be updated:

$${v_i}^{t+1} = w * {v_i}^{t+1} + c_1 * r_1 * (pbest_i^t - x_i^t) + c_2 * r_2 * (gbest_i^t - x_i^t) \tag{10}$$

$${x_i}^{t+1} = {x_i}^t + {v_i}^{t+1} \tag{11}$$

In Eq. (10), the inertia weight (w) parameter is very important and a lot of research has been done to adjust it. Constant inertia weights (w within the range [0.8, 1.2]) (50). Random inertia weights (15).

$$w = 0.5 + \frac{rand()}{2} \tag{12}$$

Eq. (12), shows linear time-varying inertia weights (15; 51; 52; 72; 73).

$$w(iter) = \frac{iter_{max} - iter}{iter_{max}}(w_{max} - w_{min}) + w_{min} \tag{13}$$

Eq. (13), shows nonlinear time-varying inertia weights (11; 19; 26; 34).

$$w(iter) = f(iter) = \{\frac{(iter_{max} - iter)^n}{(iter_{max})^n}\}(w_{max} - w_{min}) + w_{min} \tag{14}$$

Eq. (14) shows adaptive inertia weights (the inertia weight value adapted based on one or more feedback parameters) (53; 46; 68; 8; 43; 45; 57).
In (53), the inertia weight parameter is set adaptively as follows:

$$NCBPE = \frac{CBPE - CBPE_{min}}{CBPE_{max} - CBPE_{min}} \tag{15}$$

In Eq. (15), $NCBPE$: Normalized Current Best Performance Evaluation (as a performance measure of the best candidate solution found so far).
In (68), the inertia weight was adapted based on the following equation:

$$w_i^t = w_{initial} - \alpha(1 - h_i^t) + \beta s \tag{16}$$

In Eq. (16), α and β are two constants typically within the range [0,1]. h_i^t: the speed factor is define as:

$$h_i^t = |\frac{min(F(pbest_i^{t-1}), F(pbest_i^t))}{max(F(pbest_i^{t-1}), F(pbest_i^t))}| \tag{17}$$

S: aggregation degree is defined as:

$$s = |\frac{min(F_{tbest}, \bar{F}_t)}{max(F_{tbest}, \bar{F}_t)}| \tag{18}$$

In Eq. (18), $\bar{F}_t$: the mean fitness of all particles in the swarm at the $t-th$ iteration, F_{tbest} : the best fitness achieved by the particles at the same iteration.
The comfort zone parameter (c) in GOA is similar to the inertia weight (w) parameter in PSO. In this article, we use the mentioned methods and fuzzy signature to set the comfort zone parameter (c) in GOA.

3.3 Fuzzy Signatuer

Professor Lotfi Zadeh first introduced the fuzzy concept in 1965 under the title fuzzy sets and created a new and realistic worldview and approach that was more attuned to the world inhabited by humans and lent a new form to mathematics and sciences with human concepts (69). Formally, fuzzy logic can approximate a function based on linguistic input/output connections and includes three main components (i.e., inference system, a membership function, and rule bases). Today, fuzzy systems are used in a wide range of

sciences and technologies, including medicine, event prediction, and trade and commerce since fuzzy systems have a precise definition. They are a powerful instrument for modeling and controlling complex nonlinear systems, and are therefore used for defining nonlinear, unspecific, and ambiguous phenomena.
The simplified form of fuzzy multidimensional system can be considered as fuzzy signature (31). Different decision problems can be modeled with fuzzy signatures. The vector can be used to display a fuzzy signature and can be shown in Eq. (19) or a tree structure such as Fig. 7.

$$s = [[x_{11}, x_{12}]\ x_2\ [x_{31}\ [x_{321}, x_{322}, x_{323}]\ x_{33}]] \tag{19}$$

In Eq. (19), x_1, x_2, and x_3 are the members of vector x. x_1 is divided into two parts called x_{11} and x_{12}, which are shown as $[x_{11}, x_{12}]$. x_{32} is divided into three parts called x_{321}, x_{322}, and x_{323}, which are shown as $[x_{321}, x_{322}, x_{323}]$.
A variety of aggregation functions such as max, min, and mean functions can be used in fuzzy signatures. These functions must be used to reach the higher branches of the tree. For example, you can use the minimum function to convert $[x_{321}, x_{322}, x_{323}]$ to x_{32}.

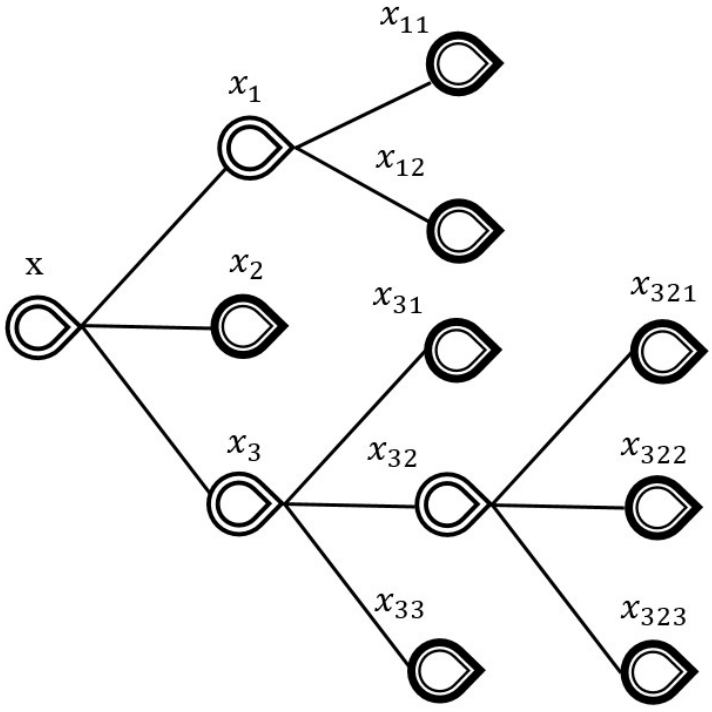

Figure 7: The tree structure of the fuzzy signature.

The advantages of fuzzy signature are expressed as follows:

- It enhances the applicability of fuzzy systems.
- It is independent of the problem properties and is suitable for complex structures.

- It can delete unnecessary information with weighted aggregated.
- It can be extracted directly from data.

4 Related works

Due to the greatness and variety of scheduling problems, research on this issue remains one of the most important issues in cloud computing. Therefore, creating a suitable and efficient algorithm is necessary.Because metaheuristic algorithms do not guarantee that the optimal solution can be found, they must be combined with other algorithms, but this increases the time complexity and is not suitable for real-world problems. To solve the mentioned NP-problem effectively, the authors in (17) suggested the Moth Search Algorithm (MSA) based on Differential Evolution (DE) for solving the task scheduling problem in cloud computing. The exploitation phase of MSA is weak, so the authors have used DE to improve this section. The authors proposed a hybrid method and named MSDE. The framework of their method consists of two phases (i.e., initial phase and updating phase). First, the best solution is selected then the DE or MSA can change the current solution. The authors simulated an improved algorithm in CloudSim. They reduced makespan.
In (2), CA is used for task scheduling. The optimized parameter is the overall response time. Tasks are scheduled based on two parameters, the first is the processing power of the machine and the second is the length of the task. In terms of makespan, comparison results have shown that the proposed cuckoo search algorithm clearly is better than the FIFO and greedy algorithms.
In (35), the task scheduling problem has been done using the bee pattern in a cloud environment. In this work, both single-objective and multi-objective parameters such as completion time, the workload of machines, and the workload of all devices are optimized. They used two types of machines, parallel and identical machines, and heterogeneous machines. As a result, the presented algorithm can improve the balance between the exploration and exploitation phases.
In (48), a hybrid of the PSO and GA is used to optimize the parameters such as completion time, scalability, and availability. To do this, two queues are used (i.e., the priority queue and queue-based on-demand). The manager stores the user tasks in the queue and then determines their priorities. After that tasks are assigned to the suitable resources. Tasks are analyzed and then stores in the queue. After that, tasks are given to the Hybrid Genetic-Particle Swarm Optimization (HGPSO) algorithm. HGPSO compared with GA and PSO. The optimized parameters are execution time, availability, and scalability.
Many studies combined optimization algorithms and taken advantage of them. In (12), the Ant Colony Optimization (ACO) algorithm is used to adjust the parameters of the PSO algorithm. This integrated algorithm can keep particles at the right level of compatibility and ensure population diversity. In addition, the best global solution with high convergence can be obtained by adjusting the learning factor. Experimental results indicated that the improved PSO algorithm could better find the fitness value based on cost

and running time.
In (41), a hybrid algorithm is used to schedule the workflow in the cloud environment. The parameters that have been optimized are the execution time and execution costs. The algorithms that are combined are the Catfish algorithm and the PSO algorithm. The simulation environment is the WorkFlowSim simulator, which is an extension of the CloudSim simulator. The effectiveness of C-PSO is determined in harsh conditions. Where the workflow is high. This algorithm improves the makespan and execution cost.
In (32) to improve the quality of service, multi-objective PSO is used to optimize parameters such as time, cost, processing power, and acceptance rate. The authors used the CloudSim simulator environment and compared it with improved PSO, Artificial Bee Community (ABC), Bat Algorithm (BA), and PSO algorithms. The simulation results show that their improved algorithm is significantly better than other advanced approaches in other conditions.
In (74), the genetic algorithm has been used to improve the completion time, the quality of work, and the average response time. The authors have used greedy algorithms and improved genetic algorithms. The novel algorithm is named MGGS. An optimal solution is found in the fewer number of iterations. The porosed algorithm is compared with GA, MGGS, MinMin, and First Come First Service (FCFS). The results proved, this algorithm improved total completion time and average response time when compared with other methods.
The most common strategies that use PSO structure suffers from the local optima problem. The task scheduling is a discrete problem but PSO suitable for continuous problems. Researchers are always looking for algorithms that are suitable for single-objective and multi-objective problems. In (3), researchers presented a discrete version of the PSO. This algorithm, called Integer-PSO, can be used to optimize one or more goals in a scheduling problem in cloud computing. When the position of the population members is updated, the values obtained are continuous. In this paper, some operators such as mod and ceil are used to overcome this problem. They proposed a new equation for position of the particles. This algorithm compared with PSO and in most cases, has had better results.
The whale optimization algorithm (WOA) suffers from the early convergence. When an algorithm has poor exploration ability, it cannot converge. In (25), researchers solved the task scheduling problem using the WOA and also improved it. By improved WOA, they optimized the execution time, response time, and operational capacity in the cloud.
One of the newest meta-heuristic algorithms, which is in the category of biological and population-based algorithms, is the Chicken swarm optimization Algorithm (CSO). This algorithm is based on the movement of chickens and their methods to search space in a population. The chickens' varied movements in finding food strike a global search. Raven roosting optimization (RRO) is in the category of biological and population-based algorithms. RRO uses individual perception mechanisms in the search process. In (61), the properties of these two algorithms are used to create a hybrid algorithm for task scheduling. The proposed method reduced runtime, response time, and increased the throughput of the cloud environment.
In (22), BA is used to solve the workflow scheduling problem. This paper focuses on

energy consumption and quality of service, so the Energy-Aware, Time, and Throughput Optimization (EATTO) algorithm is proposed. Experiments showed that EATTO can find the global optimal and optimize all three objective functions.

Table 1: Scheduling objectives of the state-of-art algorithms.

Reference	Year	Algorithm	Compared Methods	Objective Function
Elaziz et al.(17)	2019	MSA+DE	PSO WOA MSA	makespan.
Senthil et al.(48)	2018	PSO+GA	GA PSO GA PSO HEFT	completion time scalability availability.
Chen and Long.(12)	2017	PSO+ ACO	PSO ACO	makespan cost.
Nirmala and Bhanu.(41)	2016	PSO+ CATfish	PSO	makespan cost.
Kumar and Sharma(32)	2019	PSO+ Bee Colony	PSO BA ABC	makespan cost Response time.
Zhou et al.(74)	2020	Improved GA	GA Min-Min FCFS	makespan Response time.
Ajeena et al.(3)	2019	Discrete PSO algorithm (IntegerPSO)	RND-PSO SPV-PSO	makespan cost
Gu and Budati(22)	2020	EATTO based on BA	ACO Binary Search Heuristic (BSH) Random Algorithm (RN)	energy consumption execution time
Al-Zoubi(6)	2019	GOA	PSO Resource Demand Aware Scheduling (RDAS)	makespan
Adhikari et al.(1)	2020	Firefly algorithm (FA)	GSA Linewise Earliest Finish Time (LEFT)	makespan Resource utilization Reliability
Thirumalaiselvan and Venkatachalam.(59)	2019	Equal Load Balancing (ELB)+ high priority scheduling algorithm+ Rate Based Scheduling (RBS)	Cloud-DLS Adaptive EE scheduling(AEES)	Makespan average efficiency
Sheng.(49)	2019	ISACW (Improved Scheduling Algorithm for Cloud Workflow)	Max-Min DeadlineMDP	Execution time Execution cost

Table 1 shows some of the researches that have been done in the field of task scheduling in the cloud environment. Most of those approaches focus on bi-objective optimization by minimizing the cost or makespan of the task scheduling without regarding the QoS metrics such as tardiness and the load balancing of the cloud servers. This causes user

dissatisfaction. To increase the efficiency and to minimize the makespan, SGOA finds a suitable virtual machine for each task using multiple scheduling objectives (i.e., makespan, load balancing, and tardiness).

Table 2: Some types of grasshopper optimization algorithms.

Reference	Year	Category	Problem
Mafarja et al.(36)	2019	Binary GOA	Feature selection
Ewees et al.(18)	2018	Hybridization of GOA	Benchmark problems
Arora and Anand.(7)	2018	Chaotic GOA	Global optimization
Mirjalili et al.(37)	2018	Multi-objective of GOA	Multi-objective problems
Zhao et al.(71)	2019	IGOA The nonlinear comfort zone parameter was used to promote the utilization of the iterations of the algorithm	Benchmark problems
Dwivedi et al.(14)	2020	Feature Selection (EFS) + Chaotic Adaptive Grasshopper Optimization Algorithm (CAGOA) method, called ECAGOA	Feature selection

In recent years, swarm and evolutionary computing developed to optimize problems. One of the best of them is the GOA. GOA is very simple and converged in high-speed (47). Table 2 idicates some of the researches that have been done based on the grasshopper optimization algorithm. Different types of GOA have been developed for different applications (7; 14; 18; 36; 37; 71).
In this paper, we propose another extension to GOA algorithm for the task scheduling problem using fuzzy signature.

5 Task scheduling model

Definition 1: (Initial preparation time for each task). Which is marked with the symbol S_0.
Definition 2: (Preparation time between tasks). Which is marked with the symbol S. In fact, S is an $N \times N$ matrix. With this analysis, the preparation time is the same as the time required to complete the next task.
Definition 3: (The ratio of computational requirements to machine processing rate et_{ij}) (22).

$$et_{ij} = \frac{st_i}{pr_j} \tag{20}$$

In Eq. (20), st_i: the computational requirements of the $i-th$ task, and pr_j: the processing rate of the $j-th$ machine.
Definition 4: (The completion time of tasks in each machine (CTM)). We indicate the execution time on each machine with the symbol $ST_{i,j}$. To calculate $ST_{i,j}$ we have to consider S_{01} for the first task running on this machine plus the time required to run it (S_{01}+ptt_1). For subsequent tasks (e.g., $a-th$ and after that $b-th$ tasks), we have S_{ab}+ptt_b.

Fig. 9, describes this process. Suppose machine number 2 is selected to perform tasks 1 and 3, respectively, so we have $vm_2 = 1, 3$. On the other hand, the initial preparation time for these two tasks is $S_{01} = 3$,$S_{03} = 2$ and the processing time of the first is 12 ($ptt_1 = 12$) and the third task is 15 ($ptt_3 = 15$). Eq. (21) indicates the time interval between these tasks.

$$S = \begin{bmatrix} S_{11} = \infty & S_{12} = \infty \\ S_{21} = \infty & S_{22} = \infty \end{bmatrix} \tag{21}$$

According to the explanations given, because task number 1 is the first task that is performed on machine 2, so currently the time of completion of the number 2 task is the sum of two elements S_{01} and ptt_1 (Fig. 8.).

Figure 8: Calculation of the completion time of the first task on the vm_2.

After entering the number 3 task in this machine, the completion time of this machine changes as shown in Fig. 9

Figure 9: Calculation of completion time on the vm_2.

So, for each machine we have:

$$CTM_j = \sum_{i=1}^{k} (et_{i,j} + ST_{i,j}) \tag{22}$$

In Eq. (22), k: The number of tasks performed on the $j-th$ machine.
Definition 5: (makespan). The makespan of a task scheduling depends on the execution time of each task on the selected machine instance V_j (1).

$$makespan = max_{1<j<m} CTM_j \tag{23}$$

Definition 6: (The completion time of each task (CTT)). Since each task can only be performed on one machine and does not leave the machine until it is completed, the completion time, the total execution time is the time interval between this task and the previous task, and the completion time of the previous tasks. For example, the completion time of the first task in Fig. 8 is the sum of the first two parameters.
Definition 7: (Lateness). Which is characterized by Eq.24 (55).

$$lateness_i = CTT_i - d_i \tag{24}$$

Definition 8: (Unit Penalty). For each task is defined as Eq. 25 (55).

$$U_i = \begin{cases} 1, & \text{if } CTT_i > d_i \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

Definition 9: (Tardiness (T_i)). It is characterized by Eq. 26 (5).

$$T_i = max(lateness, 0) \tag{26}$$

6 Proposed comfort zone parameter adaptation

This section explains how to improve the performance of the grasshopper algorithm. Improving the performance of the grasshopper algorithm by setting the comfort zone parameter is done adaptively. The fuzzy signature has been used for adaptive adjustment. The following is a definition of the scheduling problem.

6.1 Fuzzy signature comfort zone adaptation

The fuzzy graph includes inputs, output, and aggregation functions. The output is determined using the input and aggregation functions. To indicate the value of leaves (input) in the fuzzy graph, there are two ways. The expert determines it or the value is estimated. In this paper, we consider the fuzzy signature structure as Fig. 10 and Eq. 27. We also used max, min, and mean functions as aggregation functions. The c parameter is the higher level of the structure. The adaptation of c was constrained with div_{max}, c_{max} and c_{min} where the value of div_{max} is 1, c_{min} is 0.00001, and c_{max} is 1 according to (42; 47).

$$c = [c_{max}\ [[[ps\ [div_{max}, diversity]]]\ c_{min}]] \tag{27}$$

To implement the comfort zone adaptively, we first determine the position of each particle. The percentage of success is used for this purpose. In Eq. 27, the successful count of particles is in the form of Eq. 28 (40):

$$U_i = \begin{cases} 1, & \text{if } f(p^i_{best}(t)) < f(p^i_{best}(t-1)) \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

In Eq. 28, $p^i_{best}(t)$: the best position found by particle i until iteration t, and $SC_i(t)$: the count of particle which has the best position to minimize the objective function.

$$PS(t) = \sum_{i=1}^{n_{pop}} SC_i(t) \tag{29}$$

$$PS = \frac{1}{n_{pop}} PS(t) \tag{30}$$

In Eq. 30, $PS(t)$: the percentage of success (PS) which have had an improvement in their fitness in the last iteration. The distribution of particles in the search space is considered as the diversity factor and is obtained based on Eq. 31 (42).

$$d = \sum_{i=1}^{n_{pop}} \sqrt{\sum_{d=1}^{D} (x_{id} - g_d)^2} \tag{31}$$

$$diversity = \frac{1}{n_{pop}} d \tag{32}$$

In Eq. 31, g_d: the current best particle of the swarm. x_{id}: the d-dimension of the particle i. D: the total number of dimensions.

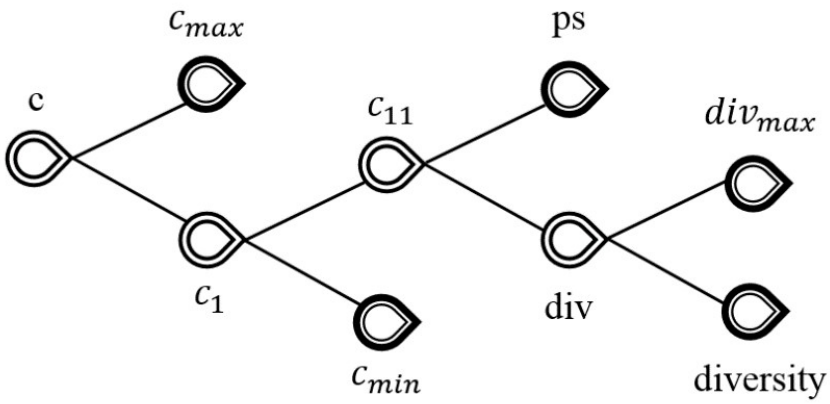

Figure 10: The structure of the fuzzy signature for the comfort zone parameter(c).

Some models may not work properly when the data is noisy, but the fuzzy signature can overcome this weakness due to bottom-up execution. In this structure, we consider c_{min} and c_{max}. The final result is in range ($[c_{min}$, $c_{max}]$). In addition to the parameters that specify the lower bound and the upper bound for parameter c, we also use div_{max}, diversity, and ps parameters to set c. In this scheme, the values of c_{min}, c_{max}, div_{max} are fixed, but the values of diversity and ps change adaptively in each iteration of the main loop. So with this interpretation, we have 100 different values for diversity and ps, which are finally converted to c using aggregation functions. Given that these parameters (diversity and ps) can be considered as membership values and their values change in the range of [0, 1].
The aggregation functions are used as follows:

$$div = div_{max} \bigcap diversity = min[div_{max}, diversity] = div_{max}\ AND\ diversity \tag{33}$$

$$c_{11} = mean[ps, div] \tag{34}$$

$$c_1 = c_{11} \bigcup c_{min} = max[c_{11}, c_{min}] = c_{11}\ OR\ c_{min} \tag{35}$$

$$c_1 = c_{max} \bigcap c_1 = min[c_{max}, c_1] = c_{max}\ AND\ c_1 \tag{36}$$

The example of the aggregation process in the structure of Fig. 10 is described as follow:
c = [1 [[[ps [1,diversity]]] 0.00001]]
Suppose that in the first iteration, the count of particles equals 21 and so the value of *ps* equals 0.21. Moreover, *diversity* equals 0.006.

$$c = [1\ [[[ps\ [1, diversity]]]\ 0.00001]] \rightarrow [1\ [[[0.21\ [1, 0.006]]]\ 0.00001]]$$
$$\xrightarrow{byEq.32} [1\ [[0.21, 0.006]\ 0.00001]]$$
$$\xrightarrow{byEq.33} [1\ [0.108, 0.00001]] \xrightarrow{byEq.34}$$
$$[1, 0.108] \xrightarrow{byEq.35} [0.108]$$

Thus, the comfort zone parameter c is 0.108.
In (47), the c parameter decreases as the iteration count increases. Adaptive update means that parameter c changes as some parameter changes from each iteration of the main loop. *Diversity* is the first parameter that is used. If in the $i-th$ iteration, the number of grasshopper around the best grasshopper is increased then the parameter c will be more in the $i+1-th$ iteration. The same scenario applies to ps parameter. Conversely, by reducing these two parameters, we will see a decrease in c parameter for the next iteration. Figures 11 , 12, and 13 describe this process.
According to Fig.13, as the comfort zone expands, it covers more grasshoppers. In this situation, the repulsive force to remove the grasshoppers prevails and on the opposite side, the grasshoppers that are outside the comfort zone are attracted to the best grasshopper with more gravity. This process causes the grasshoppers that are farther from the best grasshopper to be attracted to the best grasshopper and may better answers to be found. The opposite of this process is also possible. When the comfort zone becomes smaller, it reduces the movements of grasshoppers around the target.

6.2 The implementation of scheduling problem

Consider an environment with N tasks and M machines where tasks are independent. Each machine also has specific processing power. Assuming that the number of tasks is 4 and the number of machines is 3, a mapping of tasks and machines can be considered as follows:

$$x = \{2, 1, 2, 3\} \tag{37}$$

This means that the first task on the number 2 machine, the second task is performed on the number 1 machine, the third task is performed on the number 2 machine and the fourth task is performed on the number 3 machine (Fig. 14).

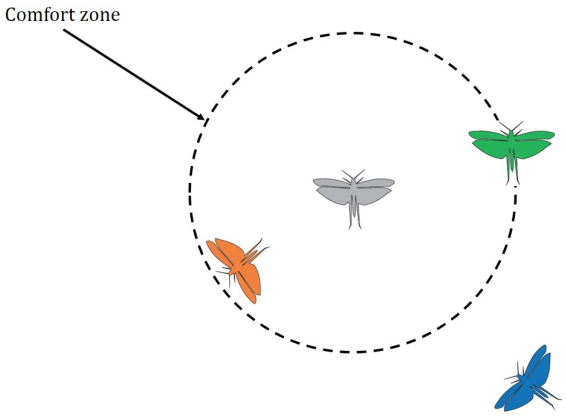

Figure 11: Comfort zone in the first iteration (47).

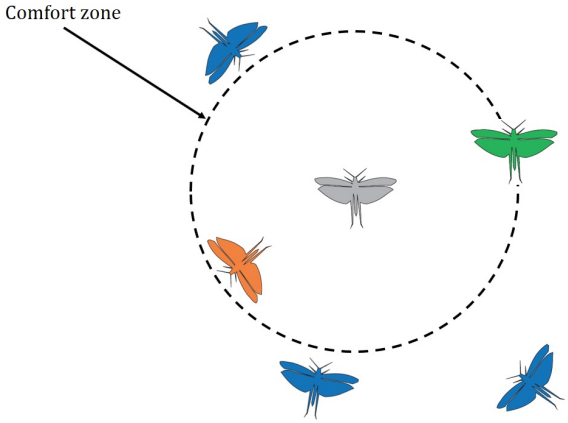

Figure 12: comfort zone in the $i-th$ iteration.

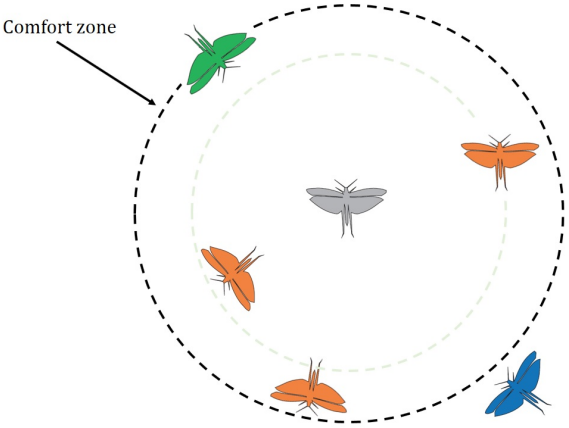

Figure 13: Comfort zone in the $i+1-th$ iteration (Provided that parameters $diversity$ and ps increase in the $i-th$ iteration) .

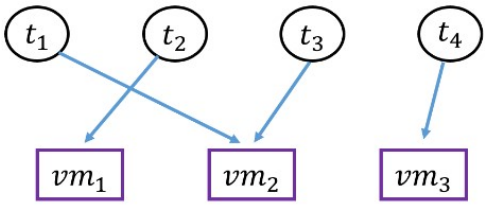

Figure 14: An example of assigning tasks to machines.

First, we determine the number of tasks and the number of machines, and then we determine the size and time of execution of the tasks, as well as the processing power of the machines. To use meta-heuristic algorithms, we need to create random solutions and so

as the x vector described earlier. We create examples and give it to meta-heuristic algorithms as a preliminary solution. Then, we try to improve these solutions and optimize the desired parameters. The general objective function is in the form of Eq. 38.

$$MinF(x) = w_1 Z1 + w_2 Z2 + w_3 Z3 \tag{38}$$

$$Z1 \longrightarrow Minf_1(x) = Max\{CTM_j\} \tag{39}$$

$$Z2 \longrightarrow Minf_2(x) = Max\{std_j\} \tag{40}$$

$$Z3 \longrightarrow Minf_3(x) = Max\{T_i\} \tag{41}$$

The parameters that are used are described in Table 3.

Table 3: Symbols and definitions.

Symbol	Definition
N	Number of tasks
M	Number of machines
st_i	The size of the $i-th$ task
$lateness_i$	The lateness of the $i-th$ task
d_i	The due date of the $i-th$ task
T_i	The Tardiness of the $i-th$ task
pt_j	processing rate of the $j-th$ machine
S_0	Initial preparation time for each task
ptt_i	The processing time of the $i-th$ task
CTT_i	The completion time of the $i-th$ task
CTM_j	The completion time on the $j-th$ machine
S_{ab}	Preparation time between tasks (N N matrix)
std_j	The standard deviation of completion time on the $j-th$ machine
et_{ij}	The size of the $i-th$ task / the processing power of the $j-th$ machine

Fig. 15 shows the scheduling system architecture. In this architecture, users send their requests to the cloud environment and wait for the results to be announced. In the cloud environment, the scheduler is obliged to select the appropriate machine based on the indicators considered in the objective function and send the tasks to the machines.
The proposed algorithm can be interpreted as Fig. 16.

7 Implementation of SGOA and Performance evaluation

In this paper, we use PSO (29), SA (30), TS (21), NSGA-II (13), and NRGA for comparison. MATLAB software has been used for simulation. First, we create several models, the number of machines is 20, and the number of tasks is 30, 50, 100 and 200. Each of the

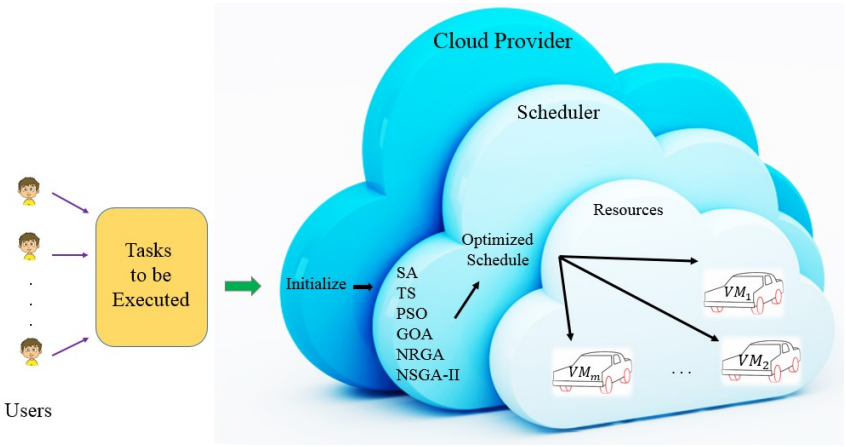

Figure 15: An Architecture of scheduling system.

Initialize the particles as Eq. (37) ;
Initialize c_{max}, c_{min}, div_{max}, *ub , lb and maximum number of iterations ;*
Evaluate particle with objective function using Eq. (38) ;
Best_particle = the best search particle ;
***While** (termination condition is false)*
    ***For** i=1 to number of particle*
        Calculate Social Interaction function ;
        ***For** d=1 to Dimension*
            Update the position of the current search particle by Eq. (9) ;
            Evaluate particle in this cube with objective function by Eq. (38) ;
            Calculate div parameter by Eq. (31) ;
        End for
        Calculate the count of particle which has the best position by Eq. (29) ;
        Update Best_particle if there is a better solution ;
    End for
    Calculate diversity parameter by Eq. (32) ;
    Calculate percentage of success (PS) by Eq. (30) ;
    Aggregate div_{max} *and* diversity *using min function resulting* ***ag1*** *;*
    Aggregate ***ag1*** *and PS using mean function resulting* ***ag2*** *;*
    Aggregate ***ag2*** *and* c_{min} *using max function resulting* ***ag3*** *;*
    Aggregate ***ag3*** *and* c_{max} *using min function resulting* ***c*** *;*
End while
Return Best_particle

Figure 16: Pseudo codes of the signature GOA.

algorithms is executed 10 times and the average results are taken and in these 10 times, the worst result and the best result are given in the Tables 9 to 12.

7.1 Comparison metric and parameter setting

The performance comparisons of the algorithms are based on speedup (64) and efficiency (23). The speedup value for a given schedule is computed by dividing the sequential execution time based on the makespan of parallel execution. It is in the form of Eq. 42.

$$speedup = \frac{serial - length}{makespan} \tag{42}$$

In Eq. 42, serial-length is computed by assigning all tasks to a single machine that minimizes execution time

$$Efficiency = \frac{speedup}{Numberofmachines} \tag{43}$$

Table 4: Symbols and definitions.

D	0.3
P	P1+P2
P1	2.05
P2	2.05
Inertia weight	c
Number of particles	50
Max number of iteration	100
Velocity of the particles	Range of VMs
Social learning factor c1	C P1
Personal learning factor c2	C P2
Inertia Weight Damping Ratio	0.1

Table 5: Adjustable parameters of SA.

T_0	10
TF	0.01
Max pre temp	10
Number of particles	50
Max number of iteration	100

Table 6: Adjustable parameters of TS.

Number of swap	N*(N-1)/2
Number of action	nSwap+ nReversion+ nInsertion
Number of insertion	N*(N-1)
Number of reversion	N*(N-1)/2
Max number of iteration	100

In Fig. 17 to 19, the average results of the algorithms are considered. Fig. 17 compares the algorithms in terms of makespan. When the size of tasks increased the makespan value also is increased. In a small number of tasks, the algorithms performe almost similarly.

Table 7: Adjustable parameters of NSGA-II.

Mutation rate	0.3
Crossover rate	0.8
Number of particles	50
Max number of iteration	100

As the number of tasks is increased, NRGA and then NSGA-II are able to complete tasks in less time. The nature of multi-objective algorithms requires that in the search space to obtain more optimal point for objective functions and individually. But among single-objective algorithms, SGOA has the best performance. SGOA in 100 tasks shows the best behaviour among all algorithms and in 200 tasks has a suitable result. It is easy to see that by using a fuzzy signature comfort zone parameter the performance of GOA can be improved and has similar or better results than the other algorithms.

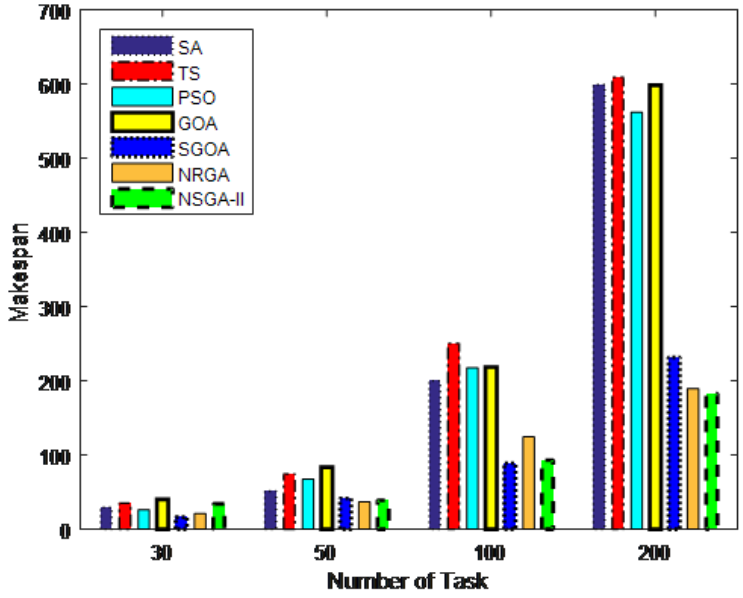

Figure 17: The comparison between the algorithms based on makespan.

In Fig. 18 and almost all of the results, the tabu search algorithm perform poorly and cannot find the appropriate points in the search space. The nature of this algorithm is similar to precise methods and so it is somewhat time consuming and cannot find the desired answer in a small number of iterations. SGOA has the best performance in load balancing between machines and creates the best balance regardless of the number of tasks.

The tardiness is a parameter that is added to the objective function in this paper. Naturally, a small amount of this parameter makes the response time shorter and thus increases the satisfaction of cloud users. Among the single-objective algorithms and for a small number of tasks (for example, 50 tasks), SGOA with the lowest latency is able to

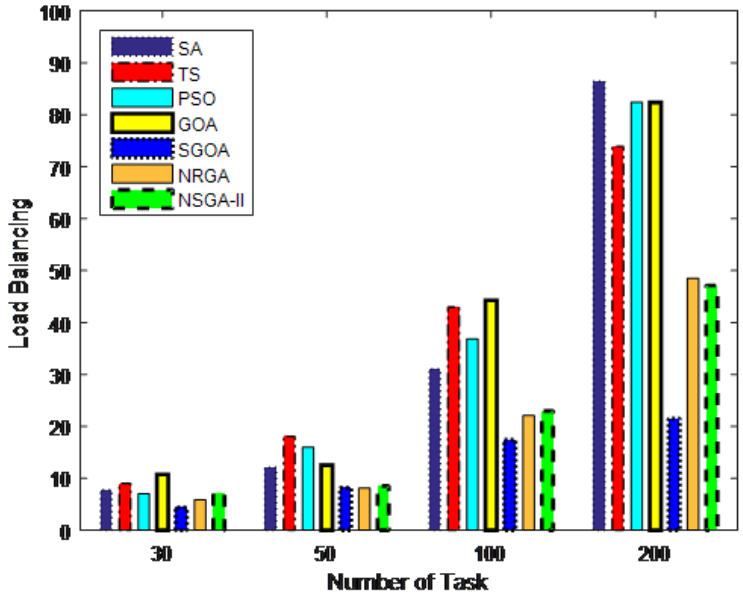

Figure 18: The comparison between the algorithms based on load balancing.

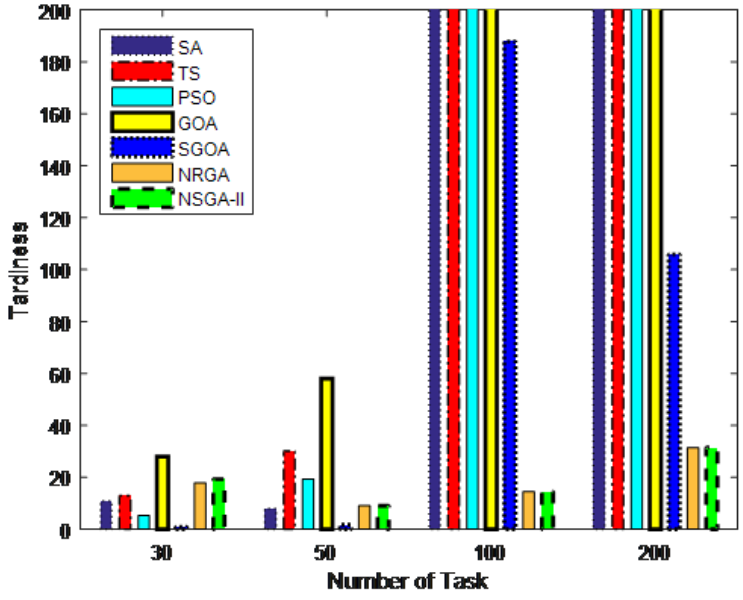

Figure 19: The comparison between the algorithms based on tardiness.

have better results than the rest of the algorithms (Fig. 19). One of the important factors of SGOA is its speed and simplicity, which has appeared in this result. However, in a large number of tasks, NSGA-II and NRGA have been able to perform scheduling with less delay.

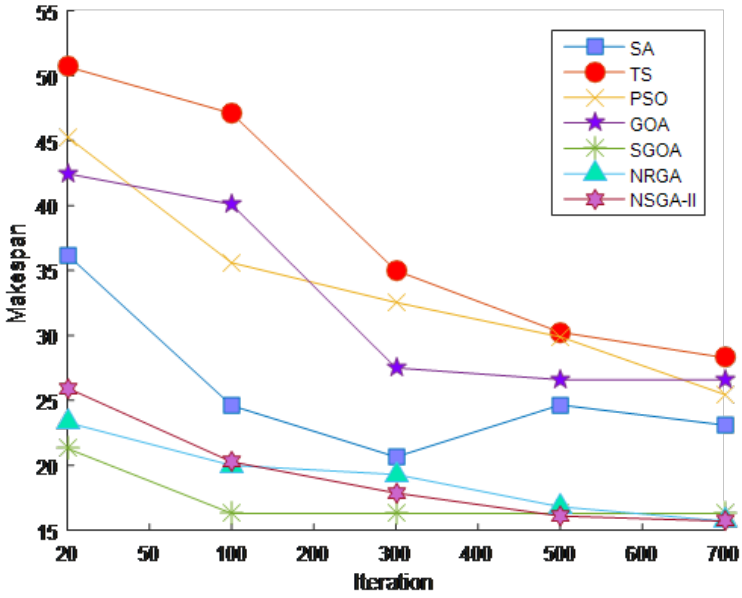

Figure 20: The comparison between the algorithms based on iteration and makespan (30 tasks and 20 machines).

The number of iterations is an effective parameter for all algorithms and is effective as a result. Fig. 20 shows the proper performance of the TS algorithm (for 30 tasks). In 20 iterations, this algorithm has a completion time of 52, but in 700 iterations, it reduces this time to 27. In this respect, it has the largest reduction compared to other algorithms. SA can achieve better makespan than the TS, PSO, and GOA. SA principally is a discrete optimization algorithm. Due to the task scheduling is a discrete problem, as expected SA has a good result. But the important point in Fig. 20 is not to change the result for SGOA. This algorithm does not change with an increasing number of iterations. In fact, the convergence speed of this algorithm has appeared with very good results in a small number of iterations, and increasing the number of iterations does not change the result.

Table 8: Comparison of algorithms based on number of particles (30 tasks - 20 machines - 20 iterations).

Number of particles	50			100			200		
Objective functions	Z1	Z2	Z3	Z1	Z2	Z3	Z1	Z2	Z3
SA	25.3	7.05	6.25	25.80	5.95	1.00	21.55	5.00	0
PSO	41.95	11.95	82.30	39.90	11.77	55.55	39.70	10.72	64.70
GOA	45.00	11.69	34.80	31.20	10.02	13.45	20.28	5.05	2.2
SGOA	15.70	5.15	0.7	15.70	4.48	0.7	14.50	4.47	0
NRGA	21.80	5.72	5.50	22.65	4.55	5.50	21.00	4.39	5.50
NSGA-II	40.62	9.64	16.83	23.95	5.03	5.50	20.15	4.67	5.50

When the population size changes, it can be seen from Table 8 that the performance

of SGOA does not change significantly. Using this feature, it determined that SGOA is suitable for doing real-time problems.

Table 9: Comparison of algorithms for scheduling 30 tasks on 20 machines based on three objective functions.

	Z1			Z2			Z3		
	Worst	Best	Mean	Worst	Best	Mean	Worst	Best	Mean
SA	36.22	23.15	29.40	9.95	6.17	7.78	18.60	3.75	10.90
TS	50.70	28.05	35.24	12.99	7.26	9.02	33.50	2.40	12.89
PSO	35.95	19.70	26.27	9.66	5.90	7.05	19.55	0	5.33
GOA	57.00	22.35	4046	14.82	7.60	10.75	53.05	7.40	27.95
SGOA	28.70	8.60	16.79	6.49	2.28	4.38	5.95	0	0.93
NRGA	37.30	16.25	21.22	7.94	3.38	5.87	56.65	5.05	17.88
NSGA-II	64.96	19.07	34.05	12.15	4.21	7.00	61.15	5.55	19.32

Table 9 shows the results of the algorithms for scheduling 30 tasks. In this number of tasks, SGOA for Z1 objective function with an average of 16.79, for Z2 objective function with an average of 4.38, and Z3 objective function with an average of 0.93 is able to answer in 41 seconds. With these interpretations, SGOA has better results than others have and can be a good option for instant and priority requests. PSO algorithm is better in terms of run-time, but because SGOA achieves the optimal answer in fewer iterations, it can be concluded that it is better. Moreover, the improvement of SGOA over others algorithm is illustrated in Table 9. When the size of tasks is 30, the performance gap between SGOA and SA, TS, PSO, GOA, NRGA, and NSGA-II is around 54.03%, 61.32%, 42.82%, 72.08%, 50.85%, and 63.39%, respectively.

Table 10: Comparison of algorithms for scheduling 50 tasks on 20 machines based on three objective functions.

	Z1			Z2			Z3		
	Worst	Best	Mean	Worst	Best	Mean	Worst	Best	Mean
SA	55.90	45.05	51.21	13.34	10.44	12.11	15.75	0.60	8.04
TS	122.74	55.40	74.57	27.19	13.80	18.07	57.55	2.75	30.12
PSO	96.70	42.10	67.50	21.32	11.30	15.98	40.55	5.80	19.37
GOA	122.45	71.70	83.55	23.76	12.56	19.11	149.50	14.80	57.97
SGOA	59.00	23.75	41.65	13.19	5.02	8.18	7.95	0	1.71
NRGA	42.85	33.01	36.74	8.84	7.39	8.09	9.04	9.04	9.04
NSGA-II	43.20	35.05	39.05	9.46	7.58	8.51	9.04	9.04	9.04

Table 10 shows the results of the algorithms for scheduling 50 tasks. In this number of tasks, the NRGA for Z1 objective function with an average of 36.74, for Z2 objective function with an average of 8.09, and SGOA for Z3 objective function with an average of 1.71, have better results. With this interpretation, the NRGA has a 20% improvement in results compared to the TS algorithm and can be a good option for this number of tasks. Among single-objective algorithms, the SGOA has better results and is very close to the NRGA. The high run-time of SGOA is compensated by reaching the answer in

a low number of iteration. When the size of tasks is 50, the performance gap between SGOA and NRGA is around 4.32%, and NSGA-II is around 9.16%.

Table 11: Comparison of algorithms for scheduling 100 tasks on 20 machines based on three objective functions.

	Z1			Z2			Z3		
	Worst	Best	Mean	Worst	Best	Mean	Worst	Best	Mean
SA	209.33	97.13	199.98	35.43	22.28	31.5	2264.65	2055.10	2145.60
TS	335.91	152.60	250.66	51.65	29.37	42.97	2680.85	2341.20	2436.21
PSO	349.86	153.05	217.68	46.71	32.16	36.85	974.70	524.20	807.78
GOA	359.39	158.15	218.66	61.41	31.76	44.27	2869.10	972.35	1575.00
SGOA	244.16	40.02	124.68	33.57	7.74	17.44	445.30	35.05	187.79
NRGA	103.41	81.40	88.99	24.51	20.23	22.09	14.53	14.51	14.51
NSGA-II	102.62	82.95	93.15	24.71	21.15	22.97	14.53	14.51	14.52

Table 11 shows the results of the algorithms for scheduling 100 tasks. In this number of tasks, the NRGA for Z1 objective function with an average of 88.99, for Z3 objective function with an average of 14.52, and SGOA for Z2 objective function with an average of 17.44, has better results. Among single-objective algorithms, SGOA has better results and is very close to the NRGA. In the three objective functions, SGOA has improved by 86.11%, 87.91%, 68.94%, 82.04% compared to algorithms SA, TS, PSO, and GOA, respectively. The poor performance of the TS algorithm can be explained by the fact that as the problem becomes more complex, this algorithm needs more time to reach the optimal solution. According to Fig. 20, TS algorithm obtains better results in more iterations.

Table 12: Comparison of algorithms for scheduling 200 tasks on 20 machines based on three objective functions.

	Z1			Z2			Z3		
	Worst	Best	Mean	Worst	Best	Mean	Worst	Best	Mean
SA	809.45	348.81	559.15	127.02	51.80	86.42	6258.30	5733.20	5881.58
TS	826.95	467.32	609.21	125.89	48.59	73.89	6860.70	5982.10	6007.36
PSO	879.89	402.51	561.95	96.81	70.64	82.43	3410.50	2482.75	3071.44
GOA	859.10	447.70	598.51	126.38	42.69	82.34	3404.70	278.30	1950.59
SGOA	425.30	125.53	231.65	40.53	11.01	21.49	297.20	0.25	105.82
NRGA	233.05	146.75	189.68	55.82	40.22	48.54	31.44	31.44	31.44
NSGA-II	213.46	153.53	182.77	51.73	41.11	47.11	31.44	31.44	31.44

Tables 11 and 12 are similar in terms of the performance of the algorithms, and the only noteworthy point is the poor performance of the PSO algorithm in 200 tasks compared to 100 tasks. When the problem became complex, PSO requires a larger initial population to achieve the desired answer. Multi-objective algorithms, both in terms of execution time and in terms of objective function optimization, still perform better than other algorithms. Among single-objective algorithms, SGOA has better results and is very close to the NRGA. In this paper, we consider 9 parameters for comparison. SGOA has

good results in 5 cases of these parameters. As the model becomes more complex, the performance of SGOA does not change much.

Table 13: Comparison of algorithms for scheduling 200 tasks on 20 machines based on three objective functions.

	30 Tasks		50 Tasks		100 Tasks	
	Speedup	Efficiency	Speedup	Efficiency	Speedup	Efficiency
SA	34.86	1.74	33.74	1.68	16.11	0.80
TS	28.85	1.44	22.86	1.14	12.51	0.62
PSO	17.79	0.88	18.37	0.91	15.96	0.79
GOA	11.19	0.55	12.05	0.60	15.80	0.79
SGOA	26.98	1.34	27.25	1.36	27.90	1.39
NRGA	32.61	1.63	20.33	1.01	24.07	1.20
NSGA-II	24.87	1.24	22.91	1.14	25.19	1.25

To identify the quality of distribution of tasks among machines, the speedup and efficiency values are estimated. Therefore, a higher value represents a more reliable result. Table 13 represents the speedup and efficiency of SA, TS, PSO, GOA, SGOA, NRGA, and NSGA-II algorithms using 20 machines. According to the experimental result, when the number of tasks is 100 the efficiency of SGOA is better than those of SA, TS, PSO, GOA, NRGA, and NSGA-II by 73.754%, 124.19%, 75.95%, 75.95%, 15.83%, and 11.20%, respectively. SA algorithm has better results in 30 and 50 tasks. When the problem becomes more complicated, the SGOA is more efficient. SGOA has better efficiency because this algorithm uses an adaptive comfort zone parameter and considers the load of tasks in machines.

8 Conclusion and future work

One of the major challenges in cloud technology is the optimal allocation of resources to tasks. Optimal scheduling has a direct effect on user satisfaction because it leads to receiving service at the proper time and at high speed. But the task scheduling problem is one of the most NP-hard problems. Due to the nature of the cloud and the number of users, the search space to find the optimal solution is very large and the allocation of resources to tasks is very time-consuming. As a result, we model the objective function based on three parameters (i.e., makespan, tardiness, and load balancing). To optimize the objective function, various optimization algorithms such as PSO (swarm-based), GA (based on evolution), SA (based on the laws of physics) and TS (based on human), are used and simulation is performed by MATLAB software environment. Multi-objective algorithms (NSGA-II and NRGA), in terms of optimization of objective functions (for example, the NSGA-II estimated the objective function to be 4% less than the SA algorithm and 5% less than the TS algorithm), has better results than the single-objective algorithms. But among single-objective algorithms, SGOA has the best performance and can optimize the objective function with a slight difference compared to multi-objective algorithms. The high execution time of SGOA is compensated by reaching the answer

in a low number of iteration. In 30 to 200 tasks, SGOA can provide acceptable performance.Since in optimization algorithms, the starting point and comprehensive search space is very effective in the result, it is suggested to use chaos theory to create diversity in the population. Adjustable parameters of the algorithm used can also be optimized with the help of tools or other algorithms to increase efficiency. In our future studies, we plan to apply machine learning techniques or artificial neural networks. In addition, the proposed method should be consider important parameters in the cloud environment such as energy, resource utilization, and reliability.

References

[1] Adhikari, M., Amgoth, T., Srirama, S. N., Multi-objective scheduling strategy for scientific workflows in cloud environment: A Firefly-based approach, Applied Soft Computing Journal, 93 (2020).

[2] Agarwal, M., Saran Srivastava, G. M., A cuckoo search algorithm-based task scheduling in cloud computing, advances in intelligent systems and computing, (2018).

[3] Ajeena Beegom, A. S., Rajasree, M. S., Integer PSO: a discrete PSO algorithm for task scheduling in cloud computing systems, Evolutionary Intelligence, (2019).

[4] Akturk, M.S., Yildirim, M.B., A new lower bounding scheme for the total weighted tardiness problem, Computers and Operations Research, 25 (1998) 265278.

[5] Akyol, D. E., Bayhan, G. M., Multi-machine earliness and tardiness scheduling problem: an interconnected neural network approach, The International Journal of Advanced Manufacturing Technology, 37 (2008) 576-588.

[6] Al-Zoubi, H., Efficient Task Scheduling for Applications on Clouds, Cyber Security and Cloud Computing (CSCloud), 6th IEEE International Conference on, IEEE, (2019).

[7] Arora, S., Anand, P., Chaotic grasshopper optimization algorithm for global optimization, Neural Comput Appl, 31 (2018) 121.

[8] Arumugam, M.S., Rao, M.V.C., On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over opera tors and root mean square (RMS) variants for computing optimal control of a class of hybrid systems, Applied Soft Computing (2008) 324336.

[9] Bozejko, W., Grabowski, J., Wodecki, M., Block approach-tabu search algorithm for single machine total weighted tardiness problem, Computers & Industrial Engineering, 50 (2006) 114.

[10] Buanga Mapetu, J. P., Chen, Zh., Kong, L., Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing, Applied Intelligence, (2019).

[11] Chatterjee, A., Siarry, P., Nonlinear inertia weight variation for dynamic adaption in particle swarm optimization, Computer and Operations Research, 33 (2006) 859871.

[12] Chen, X., Long, D., Task scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm, Cluster Comput, (2017).

[13] Deb, K., A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE transactions on evelutionary computation, 6 (2002) 182-197.

[14] Dwivedi, Sh., Vardhan, M., Tripathi, S., An effect of chaos grasshopper optimization algorithm for protection of network infrastructure, Computer Networks, (2020).

[15] Eberhart, R.C., Shi, Y.H., Tracking and optimizing dynamic systems with particle swarms, Congress on Evolutionary Computation, Korea, (2001).

[16] Edelstein-Keshet, L., Watmough, J., Grunbaum, D., Do travelling band solutions describe cohesive swarms? An investigation for migratory locusts, Journal of Mathematical Biology, 36 (1998) 515-549.

[17] Elaziz, M. A., Xiong, Sh., Jayasena, K.P.N., Li, L., Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution, Knowledge-Based Systems, 169 (2019) 3952.

[18] Ewees, A. A., Elaziz, M.A., Houssein, E.H., Improved grasshopper optimization algorithm using opposition-based learning. Expert Syst Appl, 112 (2018) 156172.

[19] Fan, S., Chiu, Y., A decreasing inertia weight particle swarm optimizer, Engineering Optimization, 39 (2007) 203228.

[20] Fisher, M.L., A dual algorithm for the one machine scheduling problem, Mathematical Programming, 11 (1976) 229252.

[21] Glover, F., future paths for integer programming and links to artificial inteligence, Computers and Operations Research, 13 (1986) 533549.

[22] Gu, Y., Budati, Ch., Energy-aware workflow scheduling and optimization in clouds using bat algorithm, Future Generation Computer Systems, 113 (2020) 106-112.

[23] Hamad, S. A., Omara, F. A., Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment, (IJACSA) International Journal of Advanced Computer Science and Applications, 7 (2016).

[24] Helo, P., Hao, Y., Toshev, R., Boldosova, V., Cloud manufacturing ecosystem analysis and design, Robotics and Computer Integrated Manufacturing, (2021).

[25] Hemasian Etefagh, F., Safi Esfahani, F., Dynamic scheduling applying new population grouping of whales meta heuristic in cloud computing, The Journal of Supercomputing, (2019).

[26] Jiao, B., Lian, Z., Gu, X., A dynamic inertia weight particle swarm optimization algorithm, Chaos, Solitons & Fractals, 37 (2008) 698705.

[27] Kamrul Hasan, S.M., Sarker, R., Cornforth, D., Hybrid genetic algorithm for solving iob-shop scheduling problem, 6th IEEE/ACIS International Conference on Computer and Information Science (IEEE ICIS), (2007).

[28] Kellegoz, T., Toklu, B., Wilson, J., Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem, Elsesier Applied Mathematics and Computation, 199 (2008) 590598.

[29] Kennedy, J., Eberhart, R. C., Particle Swarm Optimization, IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, 5 (1995) 1942-1948.

[30] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., Optimization by Simulated Annealing, Science, New Series, 220 (1983) 671-680.

[31] Kczy, L. T., Vmos, T., Bir, G., Fuzzy signatures, Proc. Eurofuse-SIC, 99 (1999) 2528.

[32] Kumar, M., Sharma, S. C., PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing, Neural Computing and Applications, (2019).

[33] Lawler, E.L., Efficient implementation of dynamic programming algorithms for sequencing problems, Report BW 106, Mathematisch Centrum, Amsterdam, (1979).

[34] Lei, K., Qiu, Y., He, Y., A new adaptive well-chosen inertia weight strategy to automatically harmonize global and local search ability in particle swarm optimization, ISSCAA, (2006).

[35] Li, J., Han, Y., A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system, Cluster Computing, (2019).

[36] Mafarja, M., Aljarah, I., Faris, H., AI. Hammouri, AlaM A-Z, S. Mirjalili, Binary grasshopper optimisation algorithm approaches for feature selection problems, Expert Syst Appl 117 (2019) 267286.

[37] Mirjalili, SZ., Mirjalili, S., Saremi, S., Faris, H., Aljarah, I., Grasshopper optimization algorithm for multi-objective optimization problems, Appl Intell, 48 (2018) 805820.

[38] Mrozek, D., A review of cloud computing technologies for comprehensive microRNA analyses, Computational Biology and Chemistry, (2020).

[39] Natesan, G., Chokkalingam, A., Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm, ICT Express 5 (2019) 110114.

[40] Nickabadi, A., Ebadzadeh, M. M., Safabakhsh, R., A novel particle swarm optimization algorithm with adaptive inertia weight, Applied Soft Computing, 11 (2011) 36583670.

[41] Nirmala, S. J., Bhanu, S. M. S., Catfish-PSO based scheduling of scientific workflows in IaaS cloud, Computing, (2016).

[42] Olivas, F., Valdez, F., Castillo, O., Melin, P., Dynamic parameter adaptation in particle swarm optimization using interval type-2 fuzzy logic, Soft Comput, 20 (2016) 10571070.

[43] Panigrahi, B.K., Pandi, V.R., Das, S., Adaptive particle swarm optimization approach for static and dynamic economic load dispatch, Energy Conversion and Management, 49 (2008) 14071415.

[44] Potts, C.N., Van Wassenhove, L.N., A branch and bound algorithm for the total weighted tardiness problem, Operations Research, 33 (1985) 177181.

[45] Qin, Z., Yu, F., Shi, Z., Wang, Y., Adaptive inertia weight particle swarm optimization, ICAISC, (2006) 450459.

[46] Saber, A.Y., Senjyu, T., Urasaki, N., Funabashi, T., Okinawa unit commitment computationa novel fuzzy adaptive particle swarm optimization approach, Power Systems Conference and Exposition, (2006) 18201828.

[47] Saremi, Sh., Mirjalili, S., Lewis, A., Grasshopper Optimisation Algorithm: Theory and application, Adv. Eng. Softw. 105 (2017) 3047.

[48] Senthil Kumar, A. M., Venkatesan, M., Task scheduling in a cloud computing environment using HGPSO algorithm, Cluster Computing, (2018).

[49] Sheng, G., Zhang, Y., Li, Y., Improved Scheduling Algorithm for Instance-Intensive Cloud Workflow, Safety Produce Informatization (IICSPI), 2nd International Conference, IEEE, (2019).

[50] Shi, Y.H., Eberhart, R.C., A modified particle swarm optimizer, IEEE International Conference on Evolutionary Computation, Anchorage Alaska, (1998) 6973.

[51] Shi, Y.H., Eberhart, R.C., Empirical study of particle swarm optimization, Congress on Evolutionary Computation, Washington DC, USA, (1999).

[52] Shi, Y.H., Eberhart, R.C., Experimental study of particle swarm optimization, SCI2000 Conference, Orlando, 2000.

[53] Shi, Y.H., Eberhart, R.C., Fuzzy adaptive particle swarm optimization, Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), Seoul, South Korea, 1 (2001) 101-106.

[54] Srinivas, M., Patnaik, L.M., Genetic algorithms: a survay, computer, 27 (1994) 17-26.

[55] Su, Sh., Yu, H., Authors Info & Affiliations Minimizing tardiness in data aggregation scheduling with due date consideration for single-hop wireless sensor networks, Wireless Networks, 21 (2015) 1259-1273.

[56] Sun, W., Zhang, N., Wang, H., Yin, W., Qiu, T., PACO: A Period ACO-based Scheduling Algorithm in Cloud Computing, International Conference on Cloud Computing and Big Data, Fuzhou, China, (2013).

[57] Suresh, K., Ghosh, S., Kundu, D., Sen, A., Das, S., Abraham, A., Inertia-adaptive particle swarm optimizer for improved global search, Eighth International Conference on Intelligent Systems Design and ApplicationsISDA, 2008.

[58] Tariq, A., Hussain, I., Ghafoor, A., A hybrid genetic akgorithm for job shop scheduling, Proceedings of the 37th International Conference on Computers and Industrial Engineering October (2007) 20-23.

[59] Thirumalaiselvan1, C., Venkatachalam, V., A strategic performance of virtual task scheduling in multi cloud environment, Cluster Comput, 22 (2019) 9589-9597.

[60] Topaz, C.M., Bernoff, A.J., Logan, S., Toolson, W., A model for rolling swarms of locusts, Eur. Phys. J. Special Topics 157 (2008) 93109.

[61] Torabi, Sh., Safi-Esfahani, F., A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing, J Supercomput, (2018).

[62] Tsujimura, Y., Mafune, Y., Gen, M., Effects of symbiotic evolution in genetic algorithms for job-shop scheduling, Proceedings of the 34th Hawaii International Conference on System Sciences IEEE, (2001).

[63] Uvarov, B., Grasshoppers and Locusts, Cambridge University Press, London, UK, 2 (1977).

[64] Verma, A., Kaushal, S., Bi-Criteria Priority Based Particle Swarm Optimization Workflow Scheduling Algorithm for Cloud, Recent Advances in Engineering and Computational Sciences (RAECS), (2014).

[65] Wen, Y., Xu, H., Yang, J., A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system, Information Sciences 181 (2011) 567581.

[66] Xing, Y., Chen, Zh., Sun, J., An improved adaptive genetic algorithm for job-shop scheduling problem, Third International Conference on Natural Computation (IEEE ICNC), (2007).

[67] Xing, Y., Wang, Zh., An improved genetic algorithm with recurrent Search for the job-shop scheduling problem, The 6th World Congress on Intelligent Control and Automation IEEE, (2006).

[68] Yang, X., Yuan, J., Mao, H., A modified particle swarm optimizer with dynamic adaptation, Applied Mathematics and Computation 189 (2007) 12051213.

[69] Zadeh, L.A, Fuzzy sets, Information and Control. 8 (1965) 338353.

[70] Zhang, L., Tong, W., Lu, Sh., Task scheduling of cloud computing based on Improved CHC algorithm, International Conference on Audio, Language and Image Processing, Shanghai, China, (2014).

[71] Zhao, R., Ni, H., Feng, H., Song, Y., Zhu, X., An improved grasshopper optimization algorithm for task scheduling problems, International Journal of Innovative Computing, Information and Control, 15 (2019) 1967-1987.

[72] Zheng, Y., Ma, L., Zhang, L., Qian, J., Empirical study of particle swarm optimizerwith an increasing inertia weight, IEEE Congress on Evolutionary Computation, (2003).

[73] Zheng, Y., Ma, L., Zhang, L., Qian, J., On the convergence analysis and parameter selection in particle swarm optimization, Proceedings of the Second International Conference on Machine Learning and Cybernetics, (2003).

[74] Zhou, Zh., Li, F., Zhu, H., Xie, H., Abawajy, J. H., Chowdhury, M. U., An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments, Neural Computing and Applications, (2020).

[75] Ziyath, S. P. M., Senthilkumar, S., MHO: meta heuristic optimization applied task scheduling with load balancing technique for cloud infrastructure services, Journal of Ambient Intelligence and Humanized Computing, (2020).