AUT Journal of Modeling and Simulation

AUT J. Model. Simul., 53(2) (2021) 197-216
DOI: 10.22060/miscj.2021.20163.5249

A Fuzzy based Pathfinder Optimization Technique for Performance-Effective Task Scheduling in Cloud

A. Zandvakili, N. Mansouri*, M. M. Javidi

Department of Computer Science, Shahid Bahonar University of kerman, Kerman, Iran

ABSTRACT: Cloud computing provides a shared pool of resources in a distributed environment and supports the features of utility-based computing. Task scheduling is a largely studied research topic in cloud computing which targets utilizing cloud resources for tasks by considering the objectives specified in QoS. Optimal task scheduling is an NP-hard problem that is time-consuming to solve with precise methods and depends on many factors, such as completion time, latency, cost, energy consumption, throughput, and load balance on the machines. Therefore, using meta-heuristic algorithms is a good selection. This paper uses the Pathfinder optimization Algorithm (PFA) for the task scheduling problem; although when the dimension of a problem is extremely increased, the performance of this algorithm decreases. In the last iterations, fluctuation rate (A) and vibration vector (ε) converg to 0, and finding a new solution is impossible. We used fuzzy logic to overcome this shortcoming and named the new algorithm Fuzzy-PFA (FPFA). In this paper, makespan, energy consumption, throughput, tardiness, and the degree of imbalance are considered as objective functions. Our goal is to minimize the makespan, energy consumption, tardiness, and degree of imbalance while maximizing throughput. Finally, different algorithms such as Firefly Algorithm (FA), Bat Algorithm (BA), Particle Swarm Optimization (PSO), and PFA are used for comparison. The experimental results indicate that the proposed scheduling algorithm can improve up to 34.2%, 16.2%, 15.9%, and 3.5% the objective function in comparison with FA, BA, PSO, and PFA, respectively.

Review History:

Received: Jun. 11, 2021
Revised: Oct. 15, 2021
Accepted: Oct. 15, 2021
Available Online: Mar. 10, 2022

Keywords:

Cloud computing
Scheduling
Fuzzy logic
pathfinder optimization.

1- Introduction

Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. Cloud computing has become the ideal way to deliver enterprise applications. This is the preferred solution for companies extending their infrastructure or launching new innovations. The goal of cloud computing is to allow the users to use applications without having them. When the user focuses on needs without considering hardware and software, it increases throughput. Companies can rent these applications instead of buying hardware and software. This reduces the cost of purchasing, repairing, maintaining, supporting, and updating. In this case, the company pays as much as it uses the resources [1].

In addition to all these benefits, cloud computing has its challenges. The task scheduling problem is one of the most important challenges [2]. The assignment of the tasks on the virtual machine is the major concern for systematic resource management [3]. One of the goals of cloud service providers is to get users' tasks done as quickly as possible. Achieving this goal requires the use of optimal task scheduling. There are several classifications of the scheduling problem in the literature. Often, scheduling problems are distinguishable based on three factors, involving machine environment, task characteristics, and objective function(s) to be optimized.

Machine environment covers single machine, multi-machine, parallel machines, uniform machines, and unrelated machines. Task characteristics cover job shop, flow shop, open shop, flexible flow shop, and flexible job-shop problems. The objective function(s) cover completion time, latency, cost, energy consumption, resource utilization, throughput, and load balancing. In this paper, machines have a different processing power, tasks are independent, and five objective functions (i.e., makespan, energy consumption, throughput, tardiness, and degree of imbalance) are optimized simultaneously..

Makespan is the time when the execution of the last task is finished [4]. Measuring the makespan is important as minimizing it will help to minimize the energy consumption. Cloud data centers consume enormous amounts of electrical energy. To support optimal servers in cloud computing, providers need to minimize cloud infrastructure energy consumption while conducting the QoS. Throughput is a measure that shows the units of information that can process in a given amount of time. Sufficient throughput ensures that tasks are properly assigned to machines [5]. Tardiness is the maximum

*Corresponding author's email: najme.mansouri@gmail.com

difference between task delivery time and expected time that is very important for user satisfaction [6]. Providers in cloud computing try to minimize this parameter. The degree of imbalance describes the amount of load distribution among the VMs, regarding their execution competencies. A low value of the degree of imbalance means that the load of the system is more balanced and efficient. All these challenges depend on the optimal scheduling [7].

Task scheduling is inherently NP-hard, and it can be very time-consuming to solve with precise methods. Therefore, one of the appropriate methods to solve it is to use meta-heuristic methods. In this paper, we use PFA to optimize objective functions. PFA is inspired by the collective movement of the animal group, and mimics the guidance hierarchy of swarms to find the best food area or hunt. This algorithm has two separate mathematical formulations for position updating of the head and other members, and can explore promising solutions [8]. However, this algorithm has some drawbacks. In the last iterations, fluctuation rate (A) and vibration vector (ε) converging to 0, and finding a new solution is impossible. To tackle this shortcoming, we use fuzzy logic to improve PFA. In our proposed model, A and ε factors are adaptively adjusted using a fuzzy inference system.

From the beginning of fuzzy concept appearance, fuzzy logic applications on the scheduling problems have increased. Fuzzy systems in the world of uncertainties provide suitable devices for the application of incorrect and qualitative data [9]. On the other hand, with the impossibility to define and combine finite mathematical functions for the changes of these parameters with finite limits, fuzzy logic is introduced as a suitable choice to adjust these parameters.

The main contributions of this paper are as follows:

- Improve PFA by the fuzzy inference system and named FPFA.
- Design a multi-objective scheduling strategy for finding an optimal mapping based on multiple conflicting objectives, namely makespan, energy consumption, tardiness, degree of imbalance, and throughput.
- Use FPFA for scheduling in the cloud computing environment.
- Use FA, BA, PSO, and PFA to compare the proposed algorithm.

The rest of this paper is organized as follows: Section 2 discusses the related works which deal with existing task scheduling techniques in the cloud environment. System architecture and problem description are discussed in section 3. The technical solution is discussed in section 4. Section 5 deals with performance evaluation and Section 6 consists of the conclusion and future work.

2- Related Work

In this section, the review of papers is divided into two parts. The first part is the application of fuzzy logic in scheduling and the second part is the application of optimization algorithms in the scheduling problem.

2- 1- Task Scheduling with Fuzzy Logic

Some papers that used fuzzy theory for task scheduling are considered in Table 1. The fact that several tasks can be performed at specific times can be presented in a table known as a timetable. The timetable is a kind of scheduling problem. The timetable for urban transportation is very complex and depends on many parameters, including the travel pattern of the passengers. In reference [10], the authors designed a timetable using a fuzzy system and came up with a new way to find the pattern of time travel between trips. The considered fuzzy variables are passenger satisfaction and vehicle capacity. Experiments have shown that their proposed system increases the load balance on buses compared to the conventional timetable, and also increases passenger comfort.

It is common to use fuzzy numbers to express the uncertainty of a phenomenon. One of the parameters considered in task scheduling is the due date, meaning that the tasks must be done at a specific time. In reference [11], the due date of the tasks is considered in the form of flexibility. They have done flexible due dates using fuzzy numbers. The authors combined the Evolutionary Algorithm (EA) and Tabu Search algorithm (TS). They named it EATS and used it to create a new neighborhood structure.

Mansouri et al. [12] proposed a hybrid algorithm based on PSO to solve task scheduling. They modified the PSO algorithm and used a fuzzy system for fitness calculations and simulated in Cloudsim [13]. The length of tasks, speed of CPU, size of RAM, and total execution time are considered fuzzy system inputs. The output of the fuzzy system is fitness. The parameters that improved are load balancing, throughput, total execution time, and makespan.

Different forms of machines and tasks can be considered for the task scheduling problem: the machines can be either the same or different, and the tasks can be either independent or dependent. In reference [14], the machines have different powers and work in parallel groups and the tasks appear in a variety of sizes. The authors minimized makespan by combining a fuzzy system and Ant Colony Optimization (ACO). They compared the proposed algorithm with the Genetic Algorithm (GA) and PSO. Experiments show that their algorithm can find better solutions than other algorithms in an acceptable time.

Fog is another layer of peripheral distributed network and is closely related to cloud computing and the Internet of Things (IoT). Fog computing is the idea of a distributed network that connects the two environments. In a short time, cloud computing can create network connections between devices and final analysis. The basis of fog computing can have different components and functions and can include fog computing gateways that collect acceptable data from IoT devices. In reference [15], the authors proposed a system based on fuzzy logic to solve task scheduling problems in the green cloud environment. When the tasks are received from IoT devices, this system selects fog or cloud environments to perform them. The proposed fuzzy system has five inputs (i.e., CPU utilization, storage utilization, bandwidth utilization, task deadline, network latency), and one output (i.e., fog

Table 1. Some scheduling algorithms based on fuzzy logic.

	Year	Method	Fuzzy logic	Compared Methods	Objective Function
[10]	2020	Designing a timetable using a fuzzy system	Number of on-board passengers as an input Passenger satisfaction as an output	Real timetable	-Load balance -Passenger comfort
[11]	2020	Doing flexible due dates using fuzzy numbers	Modeling unknown durations and flexible due dates as fuzzy numbers	Memetic algorithm (MA)	-Due-date satisfaction
[12]	2019	Solving task scheduling problem by modified PSO and fuzzy system	Length of tasks, Speed of CPU, Size of RAM, and Total execution time as inputs Fitness as an output	PSO GA	-Load balancing -Throughput -Total execution time -Makespan
[14]	2019	Applying ACO with the fuzzy number for scheduling jobs	Modeling completion time as fuzzy numbers	PSO GA	-Makespan
[15]	2021	Real-Time task scheduling based on fuzzy logic	CPU utilization, storage utilization, bandwidth, uatilization, task deadline, and network latency as inputs Fog or cloud environment as an output	SJF FIFO RTP	-Makespan -Delay rate -Average turnaround time
[16]	2021	Solving the distributed fuzzy permutation flow shop scheduling problem by ABC	Modeling processing times and due dates of the jobs as fuzzy numbers	Benchmark	-Completion time -Agreement index
[17]	2021	Combining FCFS and SJF by fuzzy rule-based scheduling method	Duration and waiting time as inputs Priority of execution as an output	FCFS SJF	-Execution time -Waiting time

or cloud environment). Makespan, delay rate, and average turnaround time are optimized and Short Job First (SJF), First In First Out (FIFO), and Real-Time Task Processing (RTP) algorithms are used for evaluation.

Baysal et al. [16] solved the distributed permutation flow shop scheduling problem with fuzzy and Artificial Bee Colony (ABC). Processing time and due dates of the jobs are considered as triangular and trapezoidal fuzzy numbers, respectively. They used 30 benchmarks and completion time with an agreement index for evaluation.

There are some heuristic-based methods for scheduling. First Come First Service (FCFS) and SJF are in this category. Each of these methods has its strengths and weaknesses, and combining these two methods and using the strengths of each can be useful. In reference [17], the fuzzy rule-based scheduling method is presented that combines these methods. The proposed fuzzy system has two inputs (i.e., duration and waiting time) and one output (i.e., the priority of execution), and is compared with FCFS and SJF to optimize the execution time and the waiting time.

Table 2. Some optimization based algorithms for scheduling

	Year	Algorithm(s)	Compared Methods	Objective Function(s)	Weaknesses
[18]	2019	Neural networks combined with reinforcement learning	-	.Makespan	. Not optimized QoS metrics . Low performance due to high time-consuming
[19]	2020	ANN and GA	-GA -MinMIN-MINMin	.Makespan .Energy consumption .Execution overhead	. High time complexity
[20]	2019	Binary PSO	HBLBA RB2B CS-PSO	.Makespan .Average waiting time .Degree of imbalance	. Comparison with only multiple versions of PSO
[21]	2021	WWO	EES HEFT	.Energy consumption .Resource utilization .Machine migration	. Not considering makespan and tardiness
[23]	2020	.LJFP-PSO .MCT-PSO	.PSO .Min-Min .Max-Min .LJFP .MCT	.Makespan .Total execution time .Degree of .Imbalance .Total energy consumption	. Low availability . High time complexity
[24]	2021	GA-WOA hybrid	.SSO .EHO .GWO	.Communication cost .Computation cost	. High time complexity . Not optimized QoS metrics . Load imbalance

2- 2- Task Scheduling with Optimization Algorithms

In this section, some papers that used optimization algorithms for scheduling are presented in Table. 2.

In reference [18], the authors proposed a new approach to tackle workflow scheduling; they applied Artificial Neural Networks (ANN) and reinforcement learning methods to create their approach and named NNS. They used two benchmark sets of workflows to test NNS. The only optimized parameter is makespan, and the authors have not compared the proposed method with any algorithm.

In reference [19], ANN is used for assigning the best resources to an arrived task in the cloud. The authors have used the ANN separation property and have trained this network with a backpropagation algorithm. Additionally, GA is used to generate a big dataset. They used GA and MinMIN-MINMin algorithms based on makespan, energy consumption, and execution overhead for evaluation. The proposed algorithm is tested in the CloudSim simulator.

In reference [20], the authors showed that meta-heuristic algorithms do not guarantee finding the optimal solution and must combine with other algorithms. However, in this case, time complexity is increased. Due to these reasons, the authors proposed a new version of binary PSO to optimize makespan, average waiting time, degree of imbalance, average resource utilization. They compared their method with Heuristic-Basedload-Balancing Algorithm (HBLBA), Range wise Busy-checking 2-way Balanced (RB2B), and Cloudlet Scheduling with Particle Swarm Optimization (CS-PSO).

Optimal use of resources is very important to reduce energy consumption in the cloud environment. In reference [21], the Water Wave Optimization (WWO) algorithm is used for this purpose, and the new algorithm is named EAS-VMC. The authors set several different goals, including energy consumption and resource utilization for optimization. To reduce energy consumption, larger tasks are separated and left to more powerful and less energy-efficient machines. The authors used real data and performed simulations in WorkflowSim environment [22]. They used Enhanced Energy-Efficient Scheduling (EES) and Heterogeneous Earliest Finish Time (HEFT) algorithms for comparison.

The starting point of the search in optimization algorithms is important to reach the optimal answer or to reach the answer in a shorter amount of time. For this purpose, Alsaidy et al. [23] used the Longest Job to Fastest Processor (LJFP) and Minimum Completion Time (MCT) algorithms to initialize the PSO algorithm. The new approaches are named LJFP-PSO and MCT-PSO. The authors used makespan, total execution time, degree of imbalance, and total energy consumption to optimize the new method compared to PSO, Min-Min, Max-Min, LJFP, and MCT.

Doing a series of preprocessing steps to schedule tasks can be helpful. In reference [24], a series of preprocessing operations are performed to improve the scheduling performance. First, the properties of tasks and machines are extracted, then these properties are reduced using Maximized the Rayleigh Quotients of the Fisher's LDA (MRQFLDA) algorithm. In the next step, the big tasks are divided into smaller and lined up tasks. Finally, the scheduling of these tasks is done using the hybrid GA- Whale Optimization Algorithm (WOA). Squieriel Search Optimization (SSO), Elephant Herd Optimization (EHO), and Grey Wolf Optimization (GWO) are selected for evaluation. The results showed that communication and computation costs are optimized.

According to the existing works, it is necessary to present a comprehensive fast algorithm with high convergence power. On the other hand, considering important parameters in scheduling simultaneous is important in customer satisfaction and service providers. For these reasons, we present an algorithm with the mentioned properties and optimize the five important parameters during the scheduling process.

3- System Architecture and Problem Description

3- 1- Problem Definition

Definition 1: (The ratio of computational requirements to processing the rate of machine ($et_{i,j}$)). It is characterized by Eq. (1) [25].

$$et_{i,j} = st_i / pr_j \tag{1}$$

in which st_i is the computational requirements of the i-th task, and pr_j is the processing rate of the j-th machine.

Definition 2: (The completion time of tasks in each machine (CTM)). We indicate the execution time on each machine with the symbol $ST_{i,j}$. Hence, we have Eq. (2) [14]:

$$CTM_j = \sum_{i=1}^{k} \left(et_{i,j} + ST_{i,j} \right) \tag{2}$$

in which k is the number of tasks performed on the j-th machine.

Definition 3: (The completion time of each task (CTT_i)). Since each task can only be performed on one machine and does not leave the machine until it is completed. CTT_i, is the time interval between i-th task and (i-1)-th task, with the addition of the completion time of the previous tasks [21].

3- 2- Objective Functions

In this section, five objective functions are introduced.

3- 2- 1- Makespan

The makespan of a task scheduling depends on the execution time of each task on the selected machine instance vm_j. It is characterized by Eq. (3) [26].

$$makespan = max_{1<j<m} \left\{ CTM_j \right\} \tag{3}$$

in which CTM_j is the completion time of tasks in the j-th machine.

3- 2- 2- Energy Consumption

The Energy Consumption (EC) is the sum of energy consumed on each selected machine. It is characterized by Eq. (4) [25].

$$EC_u = \varepsilon_u * \sum_{\forall i \epsilon u} et_{i,u} \tag{4}$$

in which EC_u is the energy consumption in the u-th machine, ε_u is the static energy consumption per time unit of the u-th machine.

$$EC = \sum_{\forall u} EC_u \tag{5}$$

in which EC is the total energy consumption.

3- 2- 3- Tardiness

Lateness is an amount of delay in executing certain operations which is characterized by Eq. (6) [27].

$$lateness_i = CTT_i - d_i \tag{6}$$

in which CTT_i is the completion time of the i-th task and d_i is the due date of the i-th task. Tardiness is a measure of a delay in executing certain operations (T_i). It is characterized by Eq. (7) [28].

$$T_i = \max\left(lateness_i, 0 \right) \tag{7}$$

3- 2- 4- Degree of Imbalance (DoI)

The degree of imbalance measures the imbalance among VMs. It describes the amount of load distribution among the VMs regarding to their execution competencies. It can be calculated as follows [29]:

$$DoI = \frac{CT_{ij}max - CT_{ij}min}{CT_{ij}avg} \tag{8}$$

in which $CT_{ij}max$, $CT_{ij}min$ and $CT_{ij}avg$ are the maximum, minimum, and average completion time of executing i-th task among total VMs, respectively.

3- 2- 5- Throughput

Throughput is calculated by finding the longest or slowest task and is calculated as follows [26]:

$$TP = 1/max\left(et_{i,j}\right) \tag{9}$$

The general objective function is in the form of Eq. (10):

$$Min\ F\left(x\right) = (w_1 .Z1 + w_2 .Z2 + w_3 .Z3 + w_4 .Z4)/(w_5 .Z5) \tag{10}$$

$$Z1\ :\ Min f_1\left(x\right) = Max\left\{CTM_j\right\} \tag{11}$$

$$Z2\ :\ Min\ f_2\left(x\right) = Max\left\{EC\right\} \tag{12}$$

$$Z3\ :\ Min\ f_3\left(x\right) = Max\left\{T_i\right\} \tag{13}$$

$$Z4\ :\ Min\ f_4\left(x\right) = Max\left\{DoI\right\} \tag{14}$$

$$Z5\ :\ Max\ f_5\left(x\right) = Max\left\{TP\right\} \tag{15}$$

Table 3 describes the parameters that are used.

Table 3. Symbols and definitions

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
pr_j	Processing rate of the *j-th* machine
ptt_i	The processing time of the *i-th* task
CTT_i	The completion time of the *i-th* task
S_0	Initial preparation time for each 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
$et_{i,j}$	The size of the *i-th* task / the processing power of the *j-th* machine

4- The Technical Solution

In this section, classic PFA is reviewed, and later the proposed algorithm (FPFA) is described.

4- 1- PFA

PFA is a new swarm-based meta-heuristic algorithm that solves optimization problems [8]. This method is inspired by the collective movement of the animal group, and mimics the leadership hierarchy of swarms to find the best food area or prey (Table 4). Avoiding local optima and achieving global optima are the features of the PFA algorithm. In this algorithm, the position of the group members and the position of the group leader are updated separately. The mathematical formulation is used for the position updating of other members that is characterized by Eq. (16) [8].

$$x_i^{k+1} = x_i^k + R_1.\left(x_j^k - x_i^k\right) + R_2.\left(x_p^k - x_i^k\right) + \varepsilon \quad (16)$$

in which k is the current iteration, x_i is the position vector of the i-th member, x_j is the position vector of the j-th member, R_1 and R_2 are the random vectors.

$$R_1 = \alpha r_1 \quad , \quad R_2 = \beta r_2 \quad (17)$$

in which α is the coefficient for interaction that defines the magnitude of the movement of any member together with its neighbor, and β is the coefficient of attraction which sets the random distance for keeping the herd roughly with the leader. In this study, α and β are randomly selected in the range of [1-2]. Additionally, r_1 and r_2 provide a random movement and are uniformly generated in the range of [0,1]. ε is for vibration that is generated in each iteration using Eq. (18).

$$\varepsilon = \left(1 - \frac{k}{k_{max}}\right) u_1.D_{ij} \quad , \quad D_{ij} = x_i - x_j \quad (18)$$

In Eq. (18), u_1 is random vectors range in [−1,1], D_{ij} is the distance between two members and k_{max} is the maximum number of iterations. To look for prey, the mathematical formulation is used for position updating of the pathfinder and is characterized by Eq. (19).

$$x_p^{k+1} = x_p^k + 2r_3.\left(x_p^k - x_p^{k-1}\right) + A \quad (19)$$

In (19), r_3 is a random vector that is uniformly generated in the range of [0,1], A is generated in each iteration using Eq. (20).

$$A = u_2 . e^{\frac{-2k}{k_{max}}} \quad (20)$$

in which u_2 is random vectors range in [−1,1].

Table 4. Pseudo code of PFA [8].

```
Load PFA parameter
Initialize the population
Calculate the fitness of initial population
Find the pathfinder
While $k < k_{max}$
    $\alpha$ and $\beta$ = random number in [1,2]
    Update the position of pathfinder using Eq. (19)
and check the bound
    If new pathfinder is better than old
        Update pathfinder
    end
    for i=2 to maximum number of populations
        Update the position of members using Eq. (16)
and check the bound
    end
    calculate new fitness of members
    find the best fitness
    If best fitness< fitness of pathfinder
        Pathfinder=best members
        Fitness= best fitness
    end
    for i=2 to maximum number of populations
        If new fitness of member(i)<fitness of member (i)
            Update members
        end
    end
    generate new A and $\varepsilon$
end
```

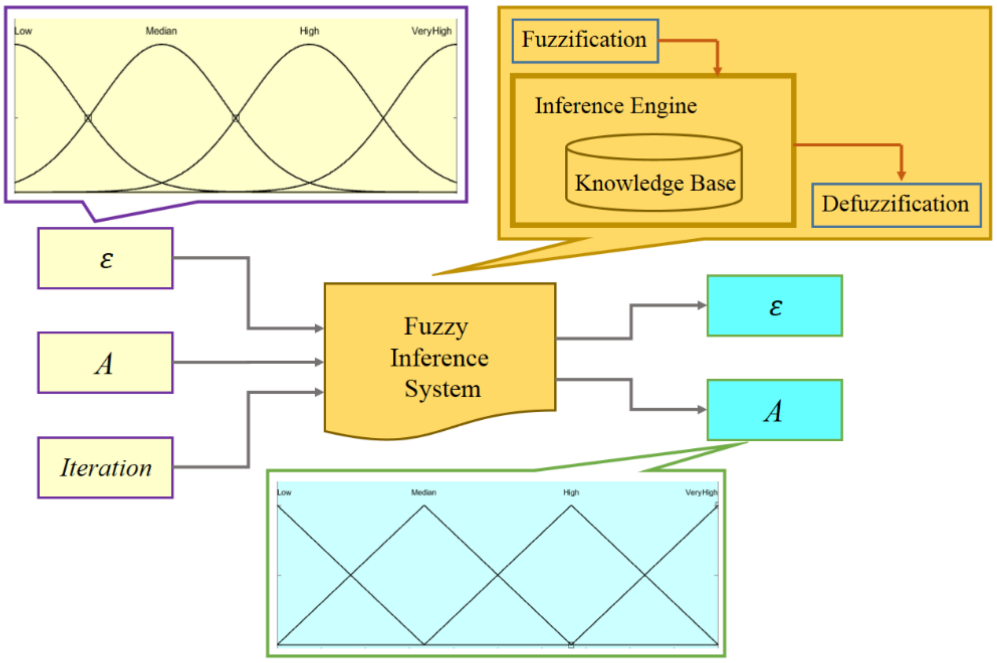


Fig. 1. Schematic of fuzzy inference system

4- 2- Proposed Fuzzy-PFA (FPFA)

Fuzzy is defined in different meanings, such as uncertain and inaccurate. In 1965, Prof. Zadeh first published the idea of fuzzy under the title Fuzzy Collections [30]. Fuzzy systems are used today in a wide range of sciences and technologies such as signal processing control, medicine, event forecasting, business, and commerce. Unlike fuzzy lexical definitions, fuzzy systems have a precise definition. Fuzzy systems are powerful tools in modeling and controlling complex nonlinear systems. Hence, these systems are used to define nonlinear and ambiguous phenomena. A fuzzy inference system consists of several parts (i.e., fuzzification, inference engine, and defuzzification). The inference engine generates the output based on the input and knowledge base. These components are shown in Fig. 1.

Along with the advantages of PFA, this algorithm also has some disadvantages. PFA can't find a new solution in the last iterations since A and ε converge to 0. To tackle this shortcoming, we used a fuzzy inference system. In the proposed method A and ε adjusted adaptively by the fuzzy inference system. In Table 5, inputs and their membership functions are introduced.

In this paper, we considered 4 membership functions for inputs and outputs. Hence, we have 4^5 rules; however, the written rules are 12. The proposed method can adjust parameters by these rules.

Table 6 shows some of the rules in the proposed system. There are some ways to design and configure fuzzy system parameters [31-32], such as using the experiences of an expert, meta-heuristic algorithms (e.g., GA [33], Bacterial evolutionary algorithm [34]), and ready-made structures such as ANFIS [35]. This article uses the knowledge and experience of an expert since there is enough knowledge available. These rules cover the weakness of PFA. In the high number of iterations, when parameters A and ε tend to 0, the fuzzy inference system prevents these parameters from becoming zero and causes exploration in the final iterations.

Table 5. Input parameters and their membership functions

Input variables	Membership functions			
ε	Low	Medium	High	Very High
A	Low	Medium	High	Very High
Iteration	Low	Medium	High	Very High

Table 6. Some examples for valid rules

Antecedents			Consequents	
ε	A	*Iteration*	ε	A
Low	Medium	Very High	Medium	Medium
Medium	Low	Very High	Medium	Medium
Low	Low	Low	High	Medium

Table 7. Input / output parameters and their characteristic

	Input			Output	
	ε	A	*Iteration*	ε	A
Range	[-1 +1]	[-1 +1]	[+1 Maxiter]	[-1 +1]	[-1 +1]
Type of membership function	Gaussian	Gaussian	Tringular	Tringular	Tringular
Number of membership function	4	4	4	4	4

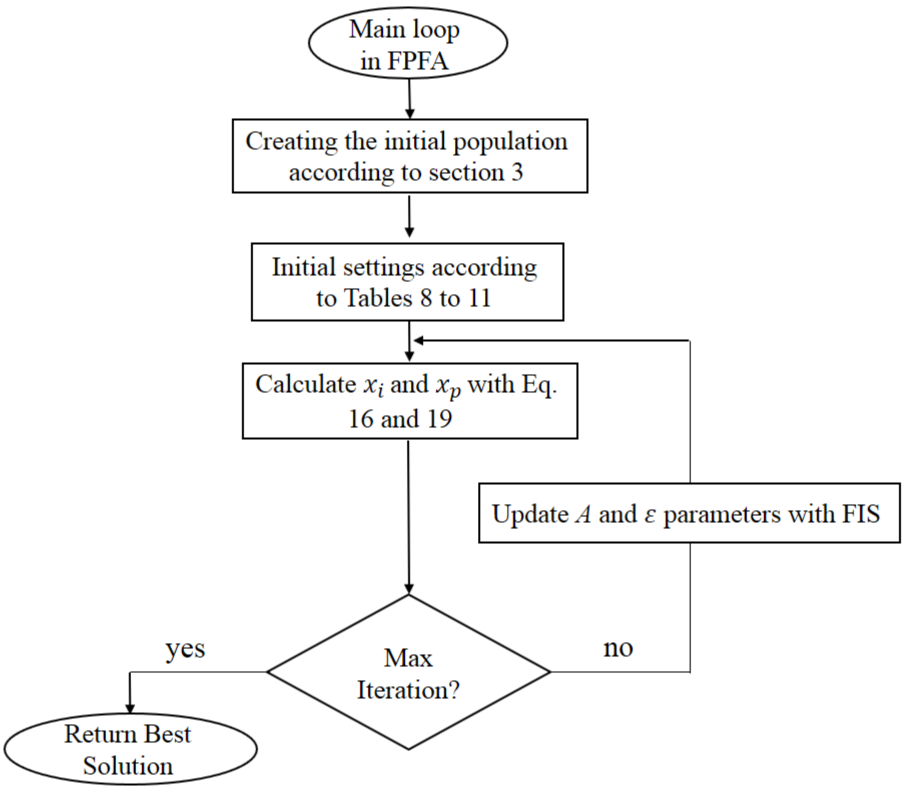

Main loop in FPFA
Creating the initial population according to section 3
Initial settings according to Tables 8 to 11
Calculate x_i and x_p with Eq. 16 and 19
Update A and ε parameters with FIS
Max Iteration?
yes
no
Return Best Solution

Fig. 2. Flowchart of FPFA

In this paper, the fuzzy inference system has three inputs and two outputs. Iteration, the values of parameters A and $\mathring{a}$ are entered as inputs, and the new A and $\mathring{a}$ are introduced as outputs. The generated outputs are used as inputs in the next iteration.

According to the classic PFA, the range of change of A and ε is in the interval [-1 +1]. The characteristic of Input/output parameters are presented in Table 7. After experiments, it is concluded that the input membership functions are of the gaussian type and the output membership functions are of triangular type.

Fig. 2 describes each iteration of FPFA. In each iteration, A and ε are adjusted adaptively by a fuzzy inference system and then these parameters are used for the next iteration.

4- 3- Task Scheduling Model

The parameters to be set are as follows:

- The number of tasks (here, N=25, 50, 70, 150)
- The number of machines (here, M=15)
- The processing power of machines
- The size and execution time of tasks

To solve the task scheduling problem using meta-heuristic algorithms, we need to create an initial population. The x vector is a sample of the population that is shown in Eq.(23).

$$\begin{cases} N = 7 \\ M = 4 \\ x = \{2,3,4,3,1,1,1\} \end{cases} \tag{23}$$

For instance, we have 7 tasks and 4 machines in Eq. (23). Corresponding to the x vector, the first task will be accepted on the second machine. The second and fourth tasks are accepted on the third machine, while the third task will be accepted on the fourth machine, the fifth, sixth, and Seventh tasks will be accepted on the first machine. In this paper, we use FA, BA, PSO, and PFA algorithms for comparison. The adjustable parameters of these methods are introduced in Tables 8 to 11.

5- Performance Evaluation

In this paper, we use the boxplot to comparison results. First, we introduce the boxplot, then use it to compare the performance of algorithms.

5- 1- Boxplot

Box diagrams provide a strong representation of data that contains useful information. A box diagram consists of five values: minimum value, first quarter, middle, third quarter, and maximum value. The density and dispersion of the data can be determined by using this diagram. The higher the data density, the more stable the algorithm is and the more reliable the results.

Table 8. Adjustable parameters of BA

Loudness	0.9
Pulse rate	0.1
Frequency minimum	0
Frequency maximum	5

Table 9. Adjustable parameters of PSO

$P1$	*2.05*
$P2$	*2.05*
P	$P1+P2$
C	$2/\left(P-2+\sqrt{P^2-4P}\right)$
D	0.3
Inertia weight	c
Inertia Weight Damping Ratio	0.1
Social learning factor c1	C$\times P1$
Personal learning factor c2	C$\times P2$
Velocity of the particles	Range of VMs

Table 10. Adjustable parameters of FA

Light Absorption Coefficient	2
Attraction Coefficient Base Value	2
Mutation Coefficient	0.2

Table 11. Adjustable parameters of PFA

Alpha	Random number in [0,1]
Beta	Random number in [0,1]

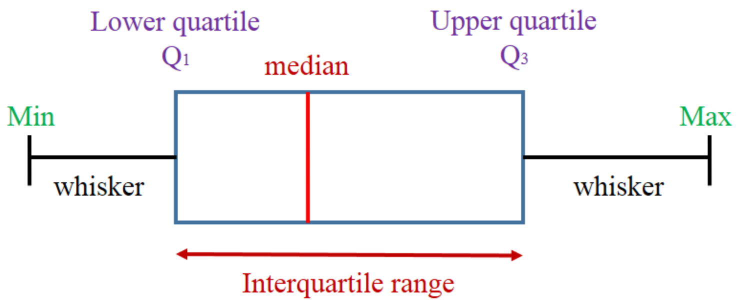


Fig. 3. Box plot structure.

Table 12. Comparison of algorithms for scheduling 25, 50, 70, 100, tasks on 15 machines based on total objective function (Z)

N	State	FA	BA	PSO	PFA	FPFA
25	Worst	1.9646e04	1.8348e04	**1.4375e04**	1.5544e04	1.4573e04
	Best	1.3164e04	1.1239e04	1.0383e04	1.0761e04	**1.0383e04**
	Mean	1.5422e04	1.3824e04	1.2769e04	1.3204e04	**1.2348e04**
50	Worst	2.6937e05	2.6570e05	1.9267e05	2.2701e05	**1.6680e05**
	Best	1.1343e05	1.1156e05	1.2465e05	1.3403e05	**1.0123e05**
	Mean	1.8734e05	1.5821e05	1.5329e05	1.7295e05	**1.4004e05**
70	Worst	1.1252e06	1.0448e06	1.0520e06	9.5599e05	**8.4250e05**
	Best	7.7238e05	5.8686e05	4.9310e05	**4.5619e05**	5.0995e05
	Mean	9.5915e05	7.8519e05	7.4278e05	6.9325e05	**6.3542e05**
100	Worst	4.0329e06	3.1830e06	3.2777e06	4.3921e06	**2.6843e06**
	Best	3.2241e06	1.5168e06	2.3050e06	**1.3139e06**	1.4288e06
	Mean	3.6999e06	2.3601e06	2.8455e06	2.3915e06	**1.9967e06**

In Fig. 3, the diagram is divided into two equal parts. The middle section, which is 50%, is known as Interquartile Range (IQR). The two sides of this section, which are 50% in total, are known as whiskers.

5- 2- Experimental Results

The stop criterion has various types. Stop criterion can be defined as a maximum number of iteration or reaching the minimum objective function [36]. The experiment was performed over 10 independent runs and 500 iterations, these parameters were chosen to be consistent and fair with other state-of-the art optimization algorithms. According to the objective functions, Z1 to Z5 are expressed in equations 11 to 15, respectively. In this section, we compare the algorithms to optimize the objective functions. The average results and the worst/best result are given in Table 12. MATLAB software has been used for simulation.

Table 12 demonstrates the results of the algorithms for scheduling 25, 50, 70, and 100 tasks. FPFA schedules 25, 50, 70, and 100 tasks with an average of 1.2348e04, 1.4004e05, 6.3542e05, and 1.9967e06 for Z objective function, respectively. FPFA can find suitable and optimal answers due to its good performance in the last iterations. The improvement of FPFA over other algorithms is illustrated in Table 12. For instance, when there are 70 tasks: the performance gap between FPFA and FA, BA, PSO, and PFA is 50.95%, 23.57%, 16.90%, and 9.10%, respectively.

The arrangement of the sections (In Fig. 4 to Fig. 7) is as follows:

Part (a): Comparison of the algorithms based on the total objective function (Z)

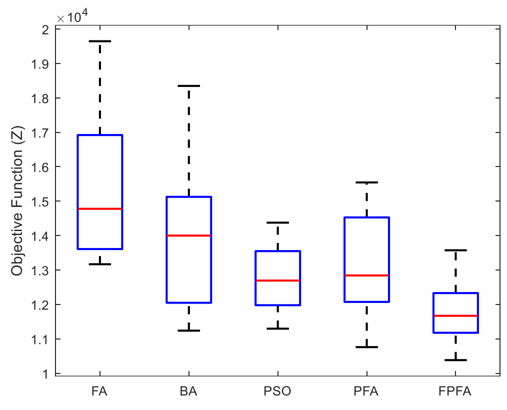
Part (b): Comparison of the algorithms based on makespan (Z1).

Part (c): Comparison of the algorithms based on energy consumption (Z2).

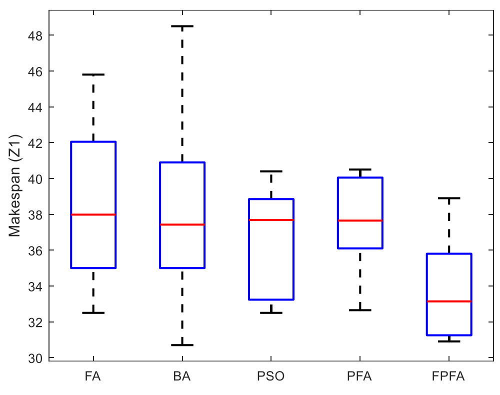
Part (d): Comparison of the algorithms based on tardiness (Z3).

Part (e): Comparison of the algorithms based on the degree of imbalance (Z4).

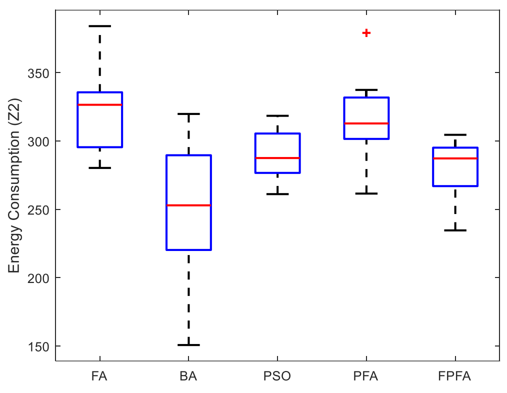
Part (f): Comparison of the algorithms based on throughput (Z5).



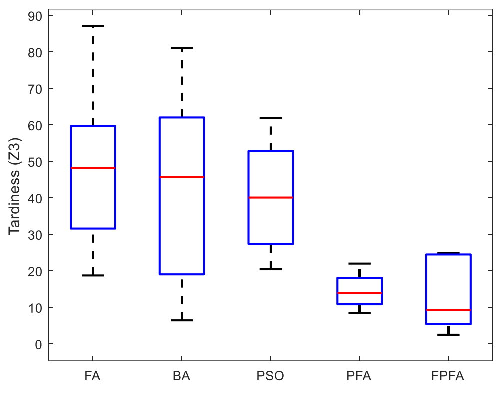
a



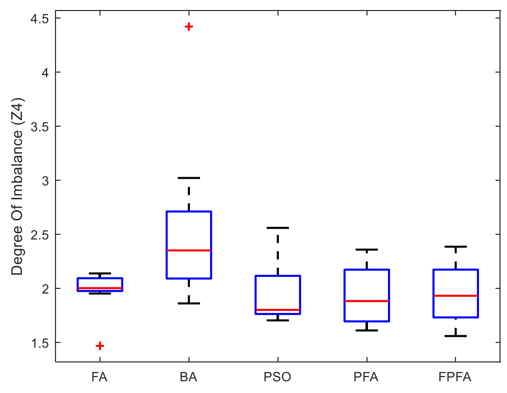
b



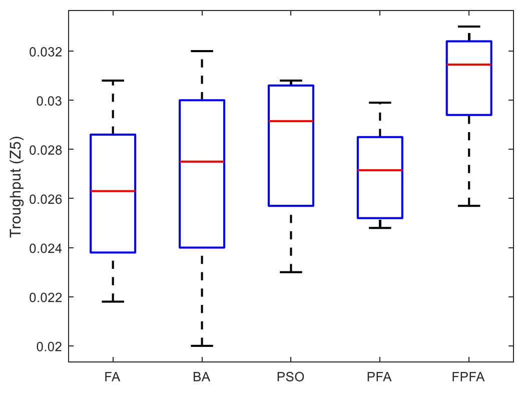
c



d



e



f

Fig. 4. a). Boxplot for total objective function (Z), b). Makespan (Z1), c). Energy consumption (Z2), d). Tardiness (Z3), e). Degree of imbalance (Z4), f). Throughput (Z5) (for 15 machines and 25 tasks).

Fig. 4 compares the algorithms when there are 25 tasks. The performance of the algorithms for the total objective function (Z) contains interesting points. In part (a) of Fig. 4, the least scatter is related to FPFA. Comparative adjustment of the effective parameters in this algorithm has led to this result. On the other hand, FPFA has the best performance in minimizing the objective function. For 25 tasks, the PSO algorithm has a suitable and acceptable performance due to its simplicity and speed. Compared to other algorithms, FA has an unbalanced performance, and the scatter of the answers obtained is good in some objective functions and is undesirable in others. For instance, the makespan objective function has a very high scatter, and it has the lowest scatter in the degree of imbalance objective function. The inability of this algorithm to exit from the local optimization implies these results.

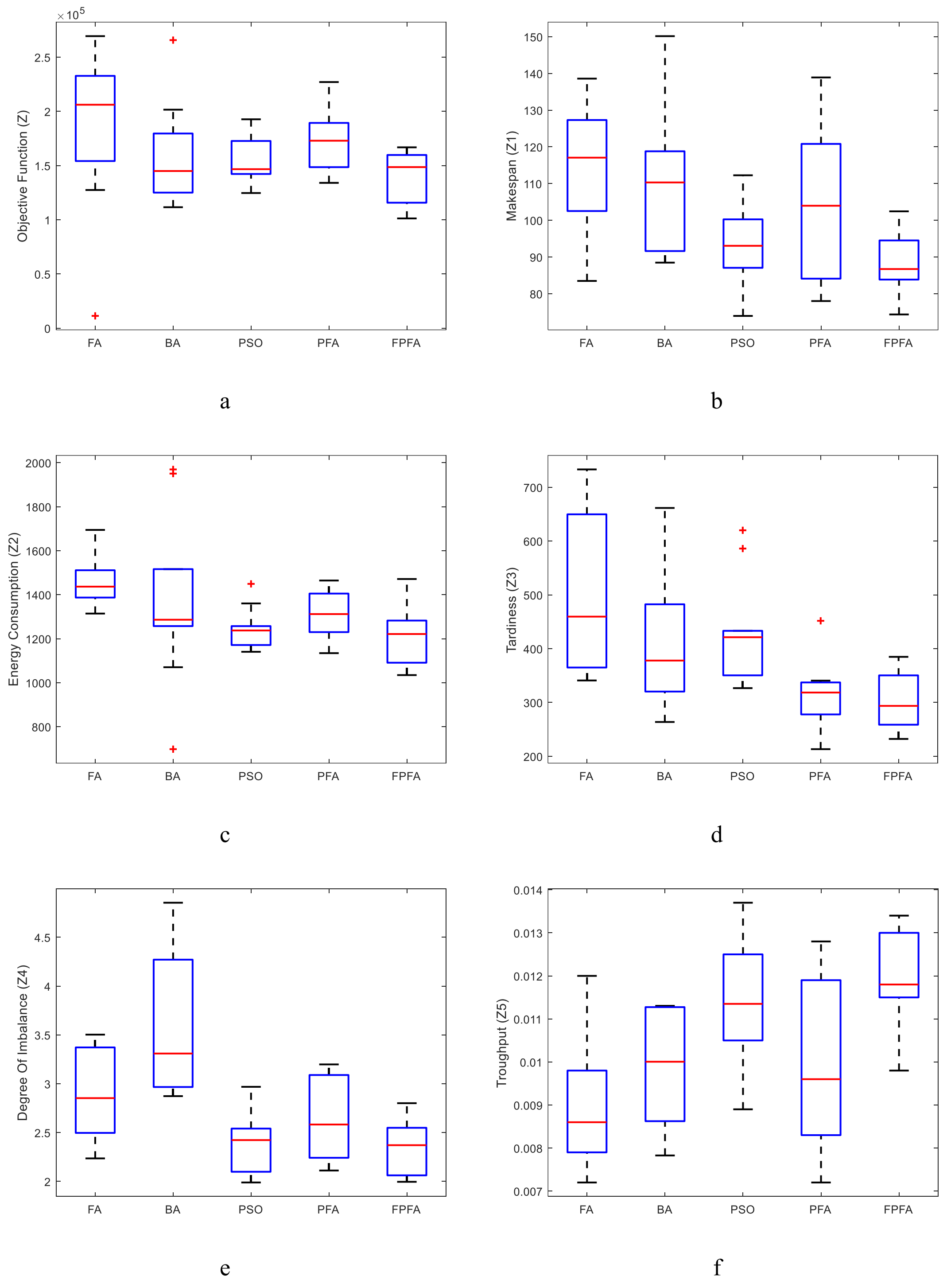


Fig. 5. a). Boxplot for total objective function (Z), b). Makespan (Z1), c). Energy consumption (Z2), d). Tardiness (Z3), e). Degree of imbalance (Z4), f). Throughput (Z5) (for 15 machines and 50 tasks).

Fig. 5 compares the algorithms when there are 50 tasks. We can see that BA has the worst performance. The inability of this algorithm to reach the global optimal has made it unable to search the problem space well and find the optimal answer. In energy consumption and tardiness objective functions, PSO algorithm has less dispersion than FPFA, but FPFA is better in minimizing the objective function. An improvement of FPFA over the PFA algorithm is evident. Adaptive adjustment of the parameters of FPFA using a fuzzy system has made this algorithm powerful. This change causes FPFA to have the power of exploration in high iterations.

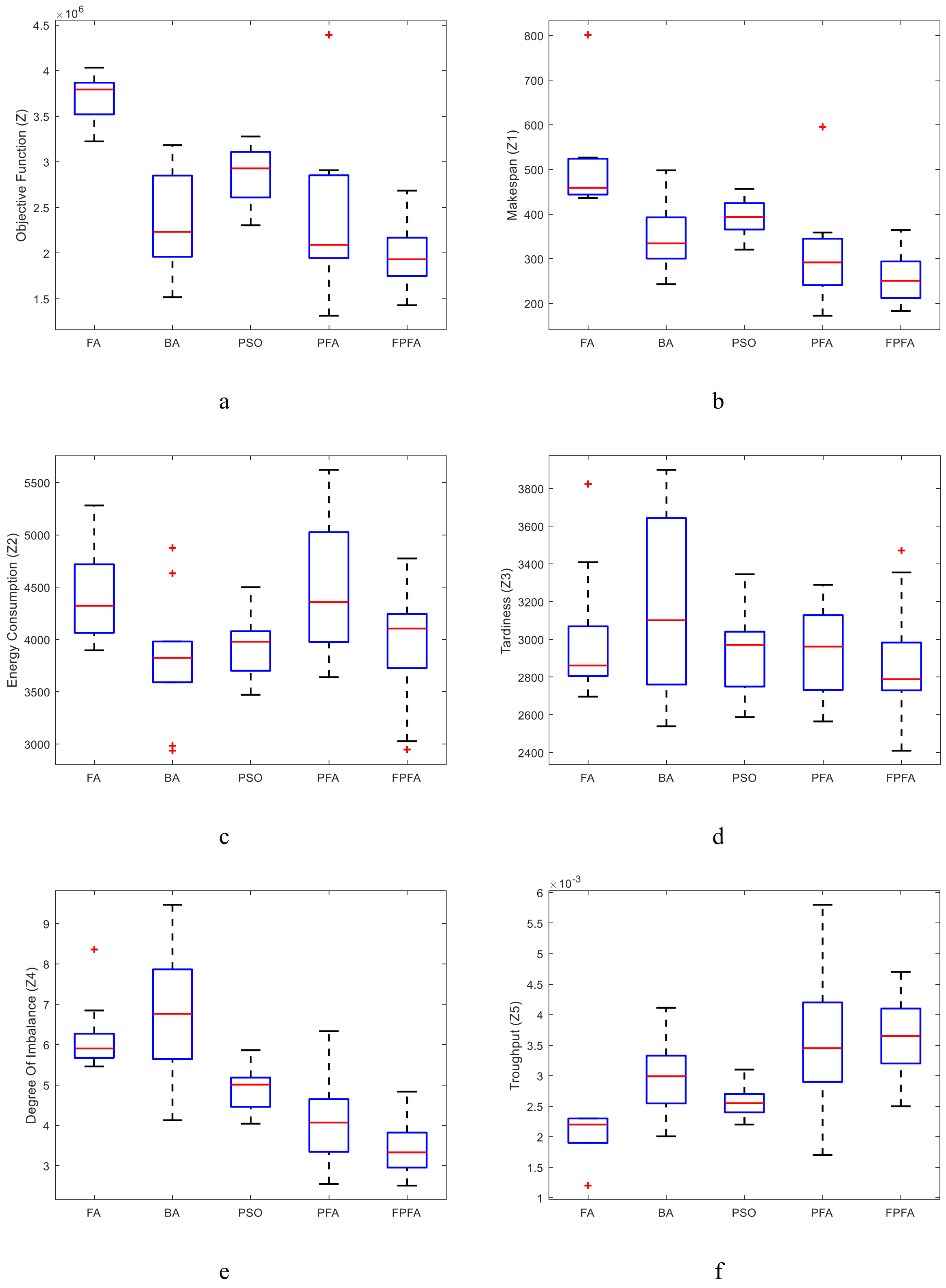


Fig. 6. a). Boxplot for total objective function (Z), b). Makespan (Z1), c). Energy consumption (Z2), d). Tardiness (Z3), e). Degree of imbalance (Z4), f). Throughput (Z5) (for 15 machines and 70 tasks).

Fig. 6 compares the algorithms when there are 70 tasks. FA performs better in minimizing the energy consumption and BA outperformed in minimizing the tardiness compared to other algorithms, but proper performance in only one target function could not be a sign of a good algorithm. The proper performance of FPFA in minimizing target functions and having less scatter is evident.

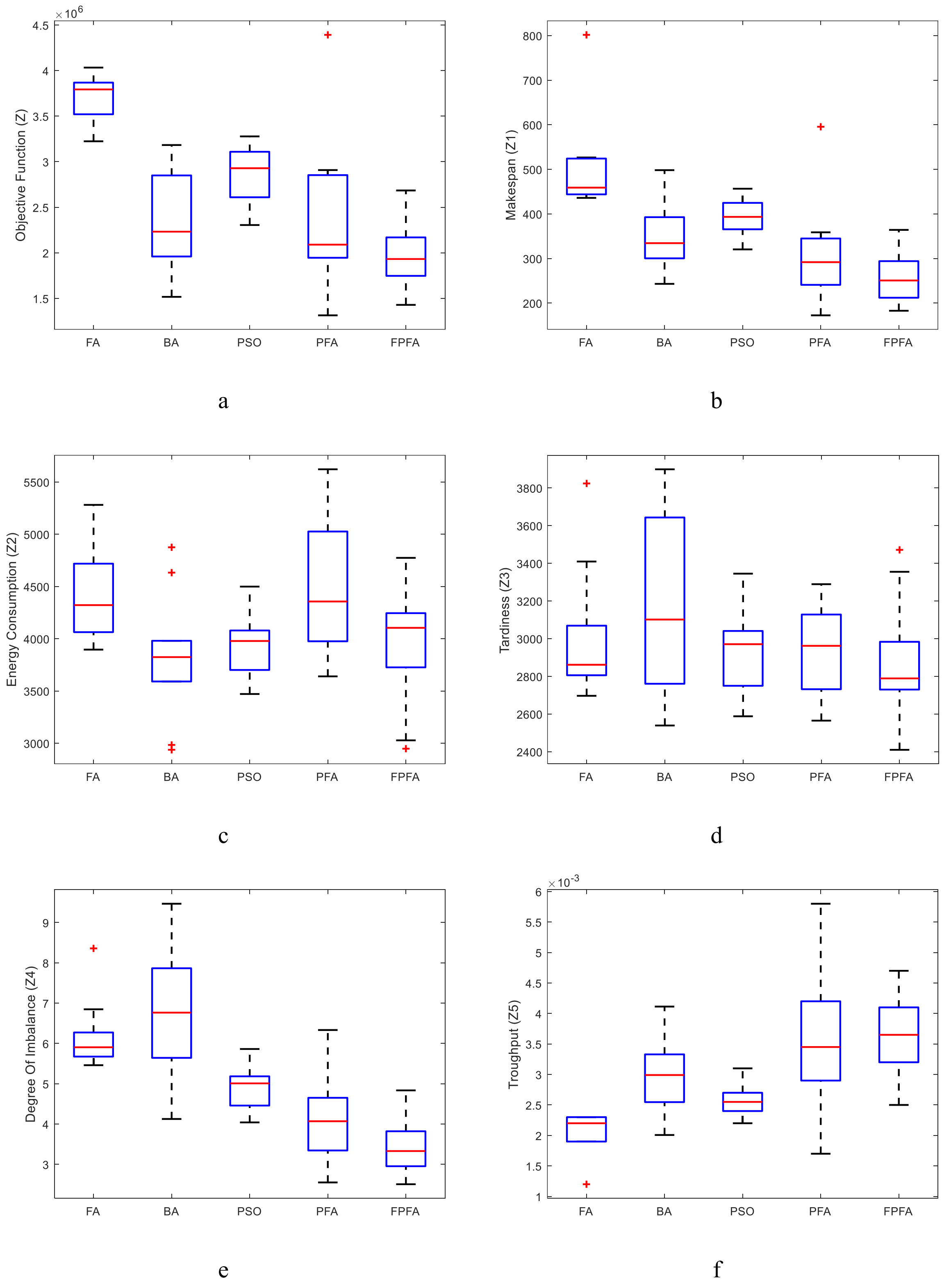


Fig. 7. a). Boxplot for total objective function (Z), b). Makespan (Z1), c). Energy consumption (Z2), d). Tardiness (Z3), e). Degree of imbalance (Z4), f). Throughput (Z5) (for 15 machines and 100 tasks).

Fig. 7 compares the algorithms for 100 tasks. When the results of an algorithm are too scattered, it becomes impossible to trust that algorithm. For example, BA performs well in minimizing energy consumption, but it also has highly scattered data. Almost all algorithms have outdated data, but the best performance in optimizing objective functions with the least scatter is related to FPFA.

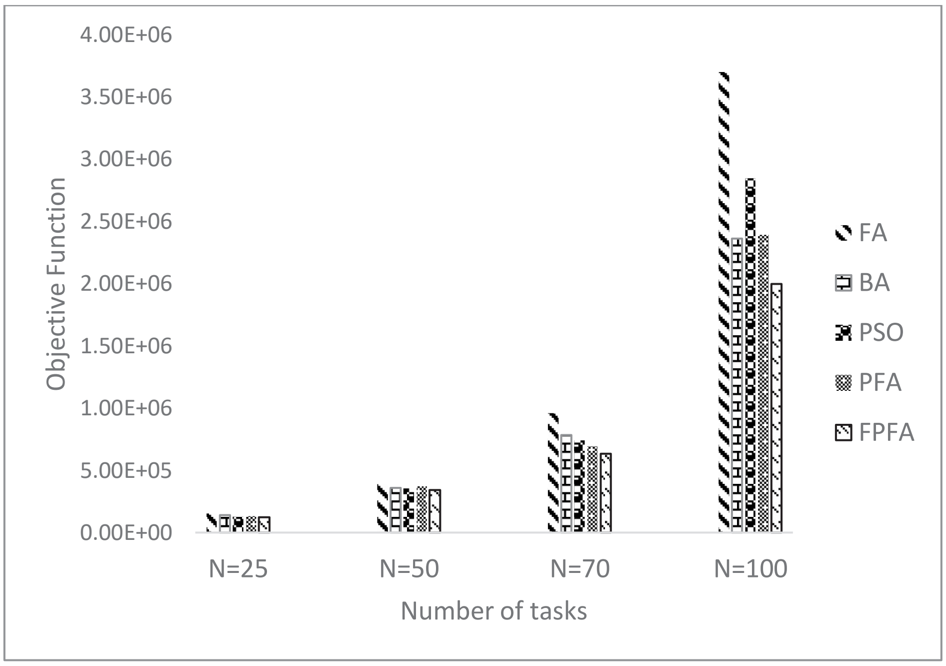


Fig. 8. The comparison between the algorithms based on objective function (Z)

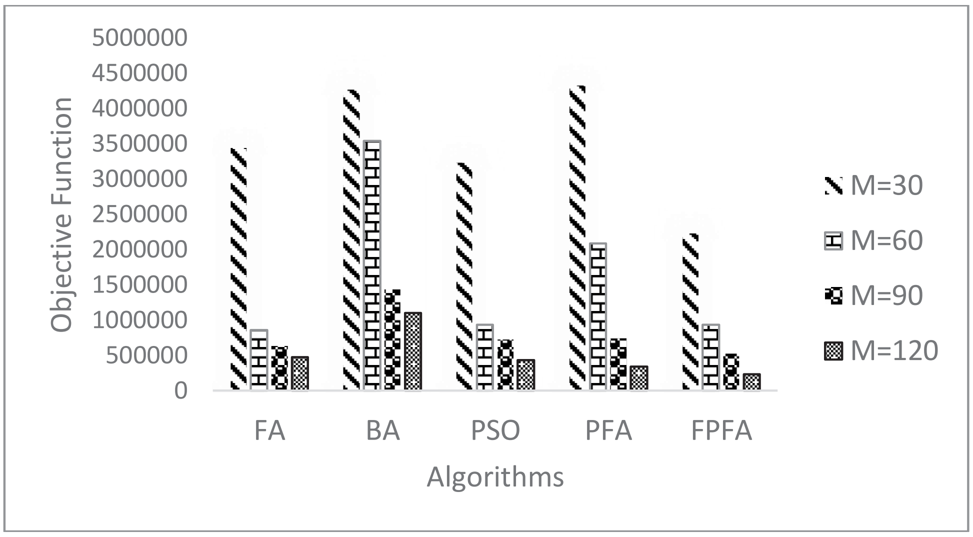


Fig. 9. The comparison between the algorithms based on objective function (Z) (number of tasks is 200 and the number of machines is 30, 60, 90, and 120).

Fig. 8 compares the algorithms in terms of the total objective function (Z). The average objective function value is gradually increased with increasing the number of tasks. When the number of tasks is low, the performance of algorithms is similar. For a large number of tasks, FPFA can improve objective function very well. For 100 tasks, in comparison to other algorithms, FPFA can improve up to 46.3%, 15.4%, 29.8%, and 16.5% the objective function in comparison with FA, BA, PSO, and PFA, respectively. By adjusting the parameters of FPFA based on the fuzzy system, the power of this algorithm in escaping from the local optimal and finding the global optimal has increased. Especially in the last iterations, it is possible to find new solutions. For these reasons, when the scheduling problem becomes complicated, this algorithm can minimize the objective function better than other algorithms.

Fig. 9 is a comparison based on the objective function (Z), provided that the number of tasks fixed at 200, and the number of machines are variable. The number of machines are 30, 60, 90 and 120. Generally, as the number of machines in-

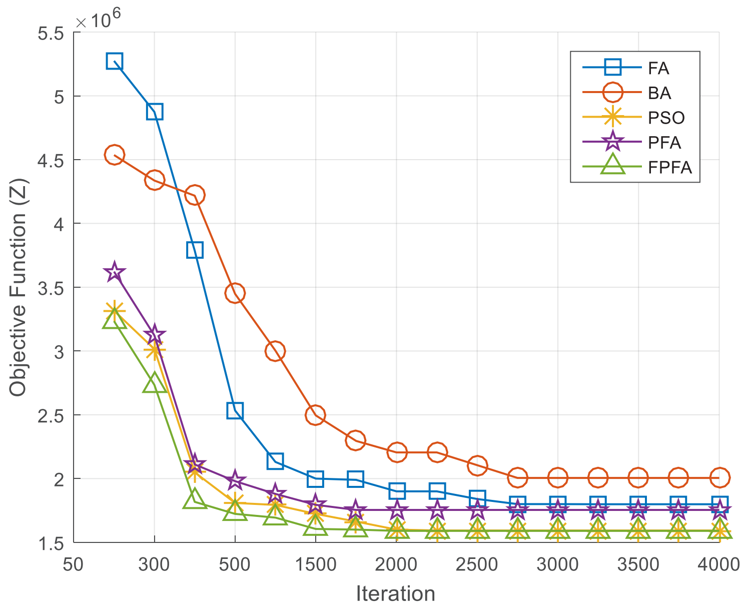


Fig. 10. Algorithms' comparison based on iteration and objective function (Z) (110 tasks and 20 machines)

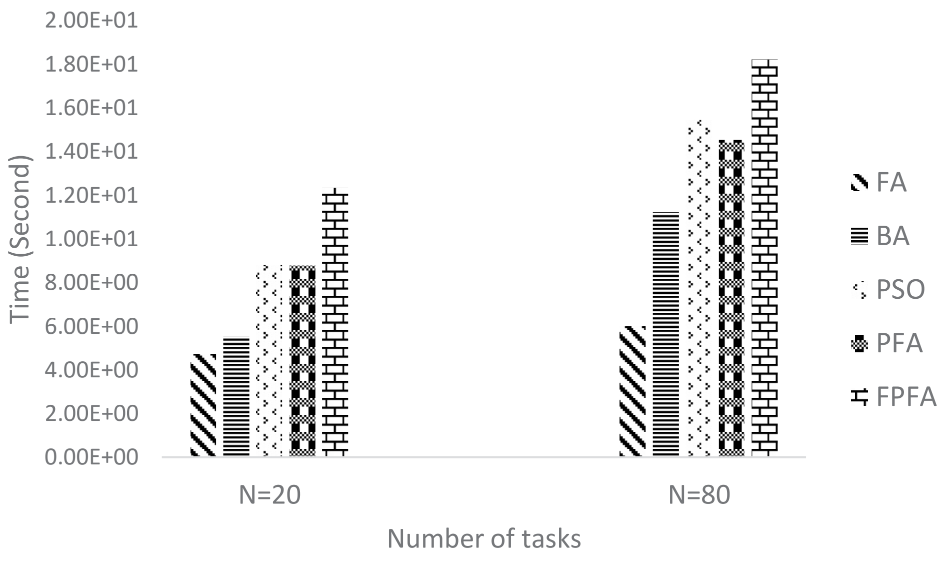


Fig. 11. Algorithms' comparison based on run-time (second)

creases, the complexity of the problem decreases. Therefore, algorithms are easier to use to solve scheduling problems. FPFA algorithm has performed well when the problem is very complex, and the number of machines is 30. In this situation, FPFA can improve up to 35.1%, 47.7%, 30.1%, and 48.3% the objective function in comparison with FA, BA, PSO, and PFA, respectively.

The number of iterations is an effective parameter for all algorithms. Fig. 10 shows the proper performance of the FA and BA algorithms. In 50 iterations, the FA algorithm has the worst state among the other algorithms, but in the final iteration, this algorithm has the suitable state. In this respect, it has the largest reduction compared to other algorithms. The best performance at the beginning and end of iterations is related to FPFA. This algorithm can be explored in the final iterations and can discover new solutions. On the other hand, this algorithm converges more quickly and achieves the desired result. It appears that FPFA is one of the top three algorithms.

Fig. 11 compares algorithms based on runtime. The number of jobs is initially 20 and then 80. The number of iterations is 500, and the size of the population is 20. FPFA has a higher runtime due to the fuzzy phase. However, this weakness can be compensated by proper convergence and achieving the desired answer in low iteration.

6- Conclusion

Allocating appropriate and distributed resources to user demands is an important issue for improving the performance of the cloud. Improper use of resources leads to wasted energy and user dissatisfaction. When tasks are assigned to suitable resources, they are completed on time and release resources. As a result, it will bring customer satisfaction. On the other hand, it increases productivity and improves the business of servers. In this paper, we modify PFA with a fuzzy inference system, named FPFA and apply this new approach to gain optimal scheduling in the cloud environment. In the new approach, the fuzzy system helps to adjust the important parameters (i.e., and A) of PFA based on the number of iterations and the previous value of these parameters. In the classical PFA, these parameters tend to zero in the high iterations, and thus losing the ability to explore. The proposed approach is used to solve the scheduling problem by considering important factors (i. e., makespan, energy consumption, tardiness, and degree of imbalance). FPFA has the best performance in minimizing makespan, energy consumption, tardiness, and degree of imbalance and also in maximizing throughput. We use other meta-heuristic algorithms (i. e., FA, BA, PSO, and PFA) for evaluation. For example, FPFA can improve up to 24.6%, 33.7%, 9.1%, and 4.3% the throughput (Z5) in comparison with FA, BA, PSO, and PFA, respectively. The condition of the problem plays an essential role in the adaptive adjustment of the parameters and causes a comprehensive search to take place. For this reason, the use of a fuzzy system creates flexibility in changing parameters and makes the algorithm more powerful. We suggest adjusting the parameters of PFA with the help of other techniques (e.g., chaos theory).

References

[1] G. Ismayilov, H.R. Topcuoglu, Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing, Future Generation Computer Systems, 102 (2020) 307-322.

[2] P.S. Rawat, P. Dimri, P. Gupta, G.P. Saroha, Resource provisioning in scalable cloud using bio-inspired artificial neural network model, Applied Soft Computing, 99 (2021) 106876.

[3] A. Ramegowda, J. Agarkhed, S.R. Patil, Adaptive task scheduling method in multi-tenant cloud computing, International Journal of Information Technology, 12(4) (2020) 1093-1102.

[4] D.K. Shukla, D. Kumar, D.S. Kushwaha, Task scheduling to reduce energy consumption and makespan of cloud computing using NSGA-II, Materials Today: Proceedings, (2021).

[5] W. Shu, K. Cai, N.N. Xiong, Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing, Future Generation Computer Systems, 124 (2021) 12-20.

[6] E.H. Houssein, A.G. Gad, Y.M. Wazery, P.N. Suganthan, Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends, Swarm and Evolutionary Computation, 62 (2021) 100841.

[7] M. Adhikari, S. Nandy, T. Amgoth, Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud, Journal of Network and Computer Applications, 128 (2019) 64-77.

[8] H. Yapici, N. Cetinkaya, A new meta-heuristic optimizer: Pathfinder algorithm, Applied Soft Computing, 78 (2019) 545-568.

[9] C.R. Miranda, F.P.V.d. Campos, M.V. Ribeiro, Energy reliability in macro base stations: A feasible solution based on a type-1 Mamdani fuzzy system, Electric Power Systems Research, 195 (2021) 107126.

[10] Y. Zhang, Q. Hu, Z. Meng, A. Ralescu, Fuzzy dynamic timetable scheduling for public transit, Fuzzy Sets and Systems, 395 (2020) 235-253.

[11] C.R. Vela, S. Afsar, J.J. Palacios, I. González-Rodríguez, J. Puente, Evolutionary tabu search for flexible due-date satisfaction in fuzzy job shop scheduling, Computers & Operations Research, 119 (2020) 104931.

[12] N. Mansouri, B. Mohammad Hasani Zade, M.M. Javidi, Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory, Computers & Industrial Engineering, 130 (2019) 597-633.

[13] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F.D. Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pract. Exper., 41(1) (2011) 23–50.

[14] Z. Jia, J. Yan, J.Y.T. Leung, K. Li, H. Chen, Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities, Applied Soft Computing, 75 (2019) 548-561.

[15] H.S. Ali, R.R. Rout, P. Parimi, S.K. Das, Real-Time Task Scheduling in Fog-Cloud Computing Framework for IoT Applications: A Fuzzy Logic based Approach, in: 2021 International Conference on COMmunication Systems & NETworkS (COMSNETS), 2021, pp. 556-564.

[16] M. Emin Baysal, A. Sarucan, K. Büyüközkan, O. Engin, Distributed Fuzzy Permutation Flow Shop Scheduling Problem: A Bee Colony Algorithm, in: C. Kahraman, S. Cevik Onar, B. Oztaysi, I.U. Sari, S. Cebi, A.C. Tolga (Eds.) Intelligent and Fuzzy Techniques: Smart and Innovative Solutions, Springer International Publishing, Cham, 2021, pp. 1440-1446.

[17] K. Chrysafiadi, A Fuzzy Task Scheduling Method, in: G.A. Tsihrintzis, M. Virvou (Eds.) Advances in Core Computer Science-Based Technologies: Papers in Honor of Professor Nikolaos Alexandris, Springer International Publishing, Cham, 2021, pp. 305-323.

[18] M. Melnik, D. Nasonov, Workflow scheduling using Neural Networks and Reinforcement Learning, Procedia Computer Science, 156 (2019) 29-36.

[19] M. Sharma, R. Garg, An artificial neural network based approach for energy efficient task scheduling in cloud

data centers, Sustainable Computing: Informatics and Systems, 26 (2020) 100373.

[20] J.P.B. Mapetu, Z. Chen, L. Kong, Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing, Applied Intelligence, 49(9) (2019) 3308-3330.

[21] R. Medara, R.S. Singh, Amit, Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization, Simulation Modelling Practice and Theory, 110 (2021) 102323.

[22] W. Chen, E. Deelman, Workflowsim: A toolkit for simulating scientific workflows in distributed environments, in: 2012 IEEE 8th international conference on E-science, IEEE, 2012, pp. 1-8.

[23] S.A. Alsaidy, A.D. Abbood, M.A. Sahib, Heuristic initialization of PSO task scheduling algorithm in cloud computing, Journal of King Saud University - Computer and Information Sciences, (2020).

[24] M.S. Sanaj, P.M. Joe Prathap, An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment, Materials Today: Proceedings, 37 (2021) 3199-3208.

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

[26] M. Adhikari, T. Amgoth, S.N. Srirama, Multi-objective scheduling strategy for scientific workflows in cloud environment: A Firefly-based approach, Applied Soft Computing, 93 (2020) 106411.

[27] S. Su, H. Yu, Minimizing tardiness in data aggregation scheduling with due date consideration for single-hop wireless sensor networks, Wireless Networks, 21(4) (2015) 1259-1273.

[28] D.E. Akyol, G.M. Bayhan, Multi-machine earliness and tardiness scheduling problem: an interconnected neural network approach, The International Journal of Advanced Manufacturing Technology, 37(5-6) (2008) 576-588.

[29] D. Alboaneen, H. Tianfield, Y. Zhang, B. Pranggono, A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers, Future Generation Computer Systems, 115 (2021) 201-212.

[30] L.A. Zadeh, Fuzzy sets, in: Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh, World Scientific, 1996, pp. 394-432.

[31] L.-X. Wang, Adaptive fuzzy systems and control: design and stability analysis, Prentice-Hall, Inc., 1994.

[32] L.S. Riza, C. Bergmeir, F. Herrera, J.M. Benítez, frbs: Fuzzy rule-based systems for classification and regression in R, Journal of statistical software, 65(6) (2015) 1-30.

[33] A. Arslan, M. Kaya, Determination of fuzzy logic membership functions using genetic algorithms, Fuzzy Sets and Systems, 118(2) (2001) 297-306.

[34] N.E. Nawa, T. Furuhashi, Fuzzy system parameters discovery by bacterial evolutionary algorithm, IEEE Transactions on Fuzzy Systems, 7(5) (1999) 608-616.

[35] M. Babanezhad, A.T. Nakhjiri, A. Marjani, M. Rezakazemi, S. Shirazian, Evaluation of product of two sigmoidal membership functions (psigmf) as an ANFIS membership function for prediction of nanofluid temperature, Scientific Reports, 10(1) (2020) 1-13.

[36] H.N. Ghafil, K. Jármai, Dynamic differential annealed optimization: New metaheuristic optimization algorithm for engineering applications, Applied Soft Computing, 93 (2020) 106392.

HOW TO CITE THIS ARTICLE

A. Zandvakili, N. Mansouri, M. M. Javidi, A Fuzzy based Pathfinder Optimization Technique for Performance-Effective Task Scheduling in Cloud, AUT J. Model. Simul., 53 (2) (2021) 197-216.

DOI: 10.22060/miscj.2021.20163.5249