IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Multi-objective Task Scheduling Approach for Fog Computing

MOHAMED ABDEL-BASSET[1], NOUR MOUSTAFA, SENIOR, IEEE,[2], REDA MOHAMED[1], OSAMA M. ELKOMY[1], and MOHAMED ABOUHAWWASH[3,4]

[1]Zagazig University, Zagazig, 44519 Ash Sharqia Governorate, Egypt (e-mails: mohamedbasset@zu.edu.eg, redamoh@zu.edu.eg)
[2]School of Engineering and Information Technology, University of New South Wales at ADFA (e-mail:nour.moustafa@unsw.edu.au).
[3]Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt (e-mail:saleh1284@mans.edu.eg).
[4]Department of Computational Mathematics, Science, and Engineering (CMSE), Michigan State University, East Lansing, MI, 48824 USA,(e-mail: abouhaww@msu.edu).

Corresponding author: Mohamed Abouhawwash (e-mail:abouhaww@msu.edu).

ABSTRACT Despite the remarkable work conducted to improve fog computing applications' efficiency, the task scheduling problem in such an environment is still a big challenge. Optimizing the task scheduling in these applications, i.e. critical healthcare applications, smart cities, and transportation is urgent to save energy, improve the quality of service, reduce the carbon emission rate, and improve the flow time. As proposed in much recent work, dealing with this problem as a single objective problem did not get the desired results. As a result, this paper presents a new multi-objective approach based on integrating the marine predator's algorithm with the polynomial mutation mechanism (MHMPA) for task scheduling in fog computing environments. In the proposed algorithm, a trade-off between the makespan and the carbon emission ratio based on the Pareto optimality is produced. An external archive is utilized to store the non-dominated solutions generated from the optimization process. Also, another improved version based on the marine predator's algorithm (MIMPA) by using the Cauchy distribution instead of the Gaussian distribution with the levy Flight to increase the algorithm's convergence with avoiding stuck into local minima as possible is investigated in this manuscript. The experimental outcomes proved the superiority of the MIMPA over the standard one under various performance metrics. However, the MIMPA couldnŠt overcome the MHMPA even after integrating the polynomial mutation strategy with the improved version. Furthermore, several well-known robust multi-objective optimization algorithms are used to test the efficacy of the proposed method. The experiment outcomes show that MHMPA could achieve better outcomes for the various employed performance metrics: Flow time, carbon emission rate, energy, and makespan with an improvement percentage of 414, 27257.46, 64151, and 2 for those metrics, respectively, compared to the second-best compared algorithm.

INDEX TERMS Multiobjective, Polynomial mutation, Cauchy distribution, fog computing, makespan.

I. INTRODUCTION

INTERNET of things (IoT) plays a crucial role in our daily real-life to enable the following changes that occur in the external world by distributing a collection of the sensors in the cities, and each one has a limited range where it could monitor the events within. After scanning the cities' changes and gathering them, they are accurately analyzed to help make the right decision. IoT could convert those cities into smart cities to improve the residents' quality of services to have better services. Due to its ability to follow the external world and help in making the right decision quickly and accurately, it is applied in several applications like healthcare [1], smart city [2], and video surveillance and emergency response [3], etc. But unfortunately, IoT devices suffer from their limited computing and storage capabilities. Thus, collecting and analyzing the data (the workload) within those devices is hard, especially the extensive data. With advancements in technology, cloud computing, with its vast storage and addressing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

TABLE 1: Nomenclature summary

Nomenclature			
$\vec{X}_L$	A vector of the lower bound values of the problem	FADs	Fish aggregating devices
$\vec{X}_U$	A vector of the upper bound values of the problem	r	A random number between 0 and 1
$\vec{X}$	Solution vector of the task scheduling	d	The number of tasks/dimensions.
E	Elite matrix	N	Population size
$\vec{R}_B$	The Brownian motion	b	Energy consumed in the active state of each VM
$\vec{X}^I$	The best-so-far solution.	si	Processing speed of each VM
t_{max}	Maximum iteration	Et	Execution time
t	Current iteration	n	Number of the virtual machines
$\vec{R}_L$	Levy flight vector	w	Workload of each task
α	Fixed-value for the levy step weight	FT	Flow time
γ	Gamma function	η_m	Index for the polynomial mutation
p_m	Mutation probability	τ	A predefined probability at the range of 0 and 1
P	Constant value	MK	Makespan
$\bigotimes$	The element by element multiplication	$E(VM_j)$	The energy consumed by each VM_j

capabilities, was used to take the IoT devices' data for its storage and analysis [4], [5], [6]. This process is known as offloading. But the cloud layer may take a long time until replying with the required data analysis due to the bandwidth overhead that occurs in the network. Subsequently, it is time for Fog computing (FC) to appear to solve the delay problem caused by using the cloud layer. FC is an intermediate layer between the cloud layer and the IoT devices near those devices until reducing the congestion in the network and improving the response time, especially for critical healthcare applications [6].

A near time ago, the FC was called edge computing, and there was no difference between them. The FC was currently used to indicate systems that provide their computing capabilities and storage services to the IoT devices via network devices such as routers and gateways. In FC, the task of data processing is performed on a fog node or IoT gateway that is located on a Local Area Network (LAN).In contrast, edge computing is a concept used to indicate the system that provides their computing capabilities and storage using the WIFI access point [3]. In edge computing, the processing is accomplished on the gateway devices that either exist closer to the data sources or on the devices with which the sensors are attached. Accordingly, the major difference between fog computing and edge computing is in terms of where data processing is accomplished.

Caching at the wireless edge is a promising manner for reducing the content request latency and the energy consumption of wireless systems by bringing the data closer to the end devices [7]. A plethora of research on caching in the wireless network has been recently done; some of which are found in [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]. We will tackle this point in detail in the future. but, in the rest of this section, more details for fog computing are discussed.

The FC consists of a number of fog nodes that are distributed in a manner to make them near the IoT devices. Afterward, the IoT devices offload the workloads to them to be analyzed. But, distributing the workloads among the fog nodes is considered a challenging task that takes the researchers' massive attention over the last period. Distributing the workloads among the fog nodes must be done to reduce the response time as much as possible until improving the quality of services for several applications, mostly autonomous driving, healthcare application, and traffic control. Moreover, the fog nodes are considered a device that consumes energy as long as it is running. Subsequently, wasting energy is regarded as the second factor that could be improved by the optimal distribution of the fog nodes' workloads. The consumed energy emits carbon dioxide that pollutes the air and harms the people in the environment. Hence, finding the optimal distribution of the workloads pays attention to the researcher nowadays to overcome the previous drawbacks resulting from the inaccurate distribution of the workloads.

Unfortunately, increasing the number of the fog nodes and the IoT devices will increase the amount of the workloads and the difficulty in finding the best assignments of those workloads to the number of the fog nodes in a manner that will reduce the carbon dioxide emission rate, response time, and energy. Generally, this problem is considered an NP-hard problem because it couldn't be solved in polynomial time by increasing the number of fog nodes and sensors, which makes the traditional methods unable to solve this problem [22]. To overcome this problem, a new attempt appeared to pay attention to using the swarm and evolutionary algorithms due to dramatic capability in solving many real-world problems in fewer time [23], [24], [25], [26], [27]. Some of the previous algorithms done to solve the task scheduling problem in FC will be reviewed within the next subsections.

In [28] the bees life algorithm (BLA) has been proposed for solving the task scheduling problem in FC (TSFC) based on attempting to achieve the optimality between the CPU time and memory storage. This algorithm dealt with those conflict objectives: the CPU time and memory storage by a weight factor which might give a preference for an objective over the other and subsequently the solution which improves the two objectives together could not be obtained in this way. Moreover, Guerrero [29] investigates the performance of three multiobjective evolutionary algorithms: weighted sum genetic algorithm (WSGA), multiobjective evolutionary algorithm based on decomposition (MOEA/D), and non-dominated sorting genetic algorithm II (NSGA-II) for finding the optimal solution for the TSFC to minimize the network delay, uses the resources optimally as possible, and allocates services. The experimental outcomes show the proficiency of both NSGA-II to optimize effectively the objectives and obtain the highest diversity of the search space and MOEA/D to minimize the execution time compared to WSGA. As elaborated in the experiments section later, that the performance of NSGA-II was so weak compared to our proposed work and this is due to the local optima problem which falls in during the optimization process.

Besides, [30] adapted the symbiotic algorithm search algorithm for optimizing the task scheduling problem for the TSFC based on the knapsack. Rahbari said that his algorithm shows improvements in the energy consumed with 18%, the execution cost with a percent of 15%, and the sensor lifetime by 5% compared with the first-come, first-served (FCFS) and knapsack algorithm. To overcome the delay and latency issues and improve the efficacy of the FC with the significant number of the requests sent to the fog and cloud, the cuckoo search algorithm (CSA) with the levy walk distribution and flower pollination (FP) ware adopted in [31] to solve those issues. This algorithm was compared with the exiting CS and the Bat algorithm [32]. The outcomes of the comparison show the superiority of CSA and FP in optimizing the response time and processing time.

Moreover, Hussein developed the ant colony optimization (ACO) and the particle swarm optimization (PSO) [33] to equilibrium the workloads among the fog nodes to minimize both response time and communication cost. Based on the comparison of ACO, PSO, and the round-robin (RR) with each other, ACO is considered the best one due to coming accurate better response times and adequate load balances among the fog nodes. However, the main limitation to this work is the energy consumption metric which didnŠt take into consideration during the optimization process, and hence this algorithm is not preferred for the TSFC because the current research are currently directed to achieving better task scheduling in less energy consumption.

Besides, to optimize resource utilization and decrease the average response time, Rafique, H., et al. [34] proposed a hybrid approach based on the modified PSO (MPSO) and modified cat swarm optimization (MCSO) algorithm to tackle this problem. MPSO was employed to schedule the fog nodes' tasks, while MCSO was used to manage the resources. The experimental results show that this hybrid approach could achieve promising outcomes for execution time, energy consumption, and average response time. Unfortunately, this hybrid algorithm didnŠt solve this problem as a multi-objective problem using the Pareto optimality strategy to tradeoff among those conflict objectives for finding the solutions which optimize all together. Moreover, a new task scheduling method [35] based on the improved firework algorithm (I-FA) has been proposed to tackle TSFC for improving both the load balancing of the fog nodes and processing time of the task as the main two objectives required to be minimized by this algorithm. I-FA was validated by two sets of experiments that show the superiority of I-FA in reducing the processing time and producing better load balancing among the fog nodes. But, it didnŠt deal with this problem using the Pareto optimality to find the optimal solutions which optimize those two conflict objectives together, but uses a weight factor to represent the significance of each objective during the optimization process.

Moreover, the Moth-flame algorithm [36] was developed for overcoming TSFC to meet the quality of service requirements of cyber-physical system (CPS) applications in a way that the total execution is significantly reduced. Comparison of MFO with some state-of-the-art algorithms affirmed its superiority with regard to the scheduling and distribution of tasks to fog nodes and the total execution time. After revising this paper, it was obvious that the authors seek for optimizing together two objectives: execution time and transfer time, which are conflict, based on a weight variable to relate the two objectives using the summation function for producing a scalar value which could be used as a fitness value. However, using this weight variable might pay attention to an objective without the other and hence the solution desired by the decision-makers might be not achieved.

Recently, a new energy-aware approach [37] based on the modified marine predators algorithm (MMPA)) has been proposed for solving the TSFC to minimize the maximum execution time until improving the response time and balancing the workloads among the fog nodes. MMPA was modified by avoiding the memory saving strategy to make the last updated solution used in the next generation even if it is not better than the local-best one. in addition, this modified variant was further improved based on two folds: the first one

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

reinitializes randomly the first half of the population to avoid stuck into local minima, and the second is based on moving the last half gradually toward the best-so-far solution to accelerate the convergence speed. This improved variant of MMPA was abbreviated as IMMPA. Ultimately, both MMPA and IMMPA were compared with four swarm-based optimization algorithms: salp swarm algorithm, whale optimization algorithm, equilibrium optimizer, and sine cosine algorithm; and the genetic algorithm, which have bad performance compared to the IMMPA under various performance metrics: makespan, energy consumption, flow time, and carbon dioxide emission rate.

In [38], the multiobjective CS (MOSCO) has been proposed for tackling the task scheduling problem for the cloud with two main objectives: the first one is minimizing the makespan, and the second one is minimizing the cloud user cost. This algorithm is compared with a number of state-of-the-art multiobjective resource scheduling algorithms, and the results of the comparison show that MOSCO is better than the others. This algorithm was compared with some relatively old and overwhelmed algorithms and its performance for some well-known and strong multiobjective optimization algorithms such as NSGA-II, and NSGA-III is not known. Another multi-objective population-based algorithm known as the crow search algorithm (CSA) has been recently developed to tackle the TSFC by improving two conflict objectives: success ratio and the security hit ratio which is maximized. In addition, this algorithm was integrated with a local search method to maximize its performance. The produced outcomes compared to some of the existing algorithms show the superiority of the hybrid CSA for tackling the TSFC. Several other works have been early done for tackling the TSFC such as hybrid heuristic and metaheuristic scheduling algorithm [39], tabu search algorithm [40], and several others [41], [42], [43].

After reviewing most of the papers published recently for tackling the TSFC, we found that they dealt with this problem as a single objective, although this problem needs to be solved by optimizing more than one objective simultaneously. For example, the best assignment of the tasks to the fog node must improve several metrics: makespan, carbon dioxide emission rate, flow time, and energy at the same time. Recently, a nature-inspired metaheuristic algorithm called marine predators algorithm has been proposed for solving the global optimization problem and could reach better solutions compared to the other nature-inspired ones for several other optimization problems such as image segmentation problem [25], parameters identification problem [44], optimal reactive power dispatch [45], and many others [46], [47], [48], [49], [50]. As a result, in this research, we propose a new multiobjective hybrid approach abbreviated MHMPA based on the marine predators' algorithm integrated with the polynomial mutation to improve the exploration capability of the algorithm within the different stages of the optimization process. Additionally, an improvement to the marine predators algorithm based on using Cauchy distribution instead of Gaussian one with the Levy Flight to give the algorithm better exploitation capability with a small ratio of the exploration at the end of the iterations until avoiding stuck into local minima is investigated. The MHMPA used the Pareto optimality to find the non-dominated solutions that will optimize the carbon dioxide emission rate and makespan at the same time. The non-dominated solutions where the improvement in any of its objectives will deteriorate at least one of the others. Moreover, we used an external archive to save the non-dominated solutions obtained within the optimization process until getting a significant number of non-dominated solutions at the end of the optimization process. The experimental results conducted to check the superiority of the proposed algorithm proved the superiority of MIMPA over the standard one and the others except for MHMPA. Therefore, the polynomial mutation is integrated with the MIMPA (MHIMPA) to check its effect on its performance. The empirical findings proved that MHMPA still is the best one, MHIMPA is the second-best one, while MIMPA is the third-best one.

The main contributions of this paper are summarized as follows:

1) Proposing a new multiobjective approach based on the MPA for solving the TSFC with two main objectives: reducing the make-span and the carbon dioxide emission rate.
2) Integrating the Polynomial mutation mechanism in an effective manner with the standard one to increase its exploration.
3) Investigating the use of Cauchy distribution with the levy flight instead of the Gaussian one on the performance of the standard algorithm until giving its better exploitation capability mixed with a small exploration capability until reducing the stuck into local minima.
4) Validating the performance of MHMPA under different task sizes with heterogeneous workloads and different VMs lengths.
5) Comparing the performance of MHMPA with a number of the well-known robust multiobjective algorithms to check its superiority. Based on this comparison, our proposed was the best under four used metrics: Make-span, carbon emission rate, Flow Time, and energy.

Finally, the rest of this paper is arranged as follows:

1) Section II overviews the standard marine predators' algorithm.

2) Section III introduces the computational model for the algorithm.
3) Section IV explains our proposed algorithm (MIMPA).
4) Section V exposes the outcomes and compares them with a number of the algorithms.
5) Section VI summarizing the experiments.
6) Section VII shows our conclusions about this work with our future work.

II. OVERVIEW OF MARINE PREDATORS ALGORITHM.

Recently, a new meta-heuristic algorithm [51] based on the predators' nature that follows specific behaviors for catching their prey has been proposed for tackling the optimization problems. This algorithm is known as the marine predators' algorithm (MPA). For finding and attacking the prey, the predators use two strategies: Brownian and Levy. In the Brownian approach, the step size between the predators and their prey will be enormous. This step is performed at the start of the optimization process to cover the optimization problem's search space as much as possible. At the same time, Levy's strategy is applied when the optimization process is so near the end to focus on the best-so-far solution to increase the convergence toward it as an attempt to find a better solution. At the start of the optimization process, a set consisting of N solutions will be randomly distributed within the search space of the problem according to the following formula:

$$\vec{X} = \vec{X}_L + \vec{r}.(\vec{X}_U - \vec{X}_L) \tag{1}$$

$\vec{r}$ is a vector involving values generated randomly between a value greater than or equal to 0 and less than 1. $\vec{X}_L$ is a vector proposed to contain the lower bound of each dimension of the optimization problem, while $\vec{X}_U$ indicates each dimension's upper bound. After distributing the solution within the problem's search space, the fitness value of each one will be calculated, and the one with the best fitness value, if the problem is minimized, then the best will be the one with the lowest fitness values. Otherwise, the one with the highest fitness value will be the best. According to the survival of the fitness theory, the solution with the best fitness is considered the better one for the foraging process, so it will be used to construct a matrix known as Elite, abbreviated as E.

$$E = \begin{bmatrix} X^I_{1,1} & X^I_{1,2} & \dots & X^I_{1,d} \\ X^I_{2,1} & X^I_{2,2} & \dots & X^I_{2,d} \\ \dots & \dots & \dots & \dots \\ X^I_{N,1} & X^I_{N,2} & \dots & X^I_{N,d} \end{bmatrix}$$

X^I is the best-so-far solution and duplicated N times to construct the E matrix, and d is the dimensions length of the optimization problem. The elite matrix is of a size $N \times d$, where each row in this matrix will be used for searching for the prey based on the information on the preyŠs positions py. After that, the predators will be moved toward the preyŠs positions that are formulated in the following matrix of size $N \times d$:

$$py = \begin{bmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,d} \\ X_{2,1} & X_{2,2} & \dots & X_{2,d} \\ \dots & \dots & \dots & \dots \\ X_{N,1} & X_{N,2} & \dots & X_{N,d} \end{bmatrix}$$

After building the E and py matrices, the optimization process will start to simulate the behaviors of the predators during searching for their prey. At the start of the optimization process, when the prey is moving faster to the predators, the best strategy for the predators doesnŠt move at all since the prey will come themselves. This is called an exploration phase, where the prey will explore the search space for finding their foods. It is worth mentioning that predators will search for their prey while those prey will search for their food within the search space and subsequently the predators and prey are considered as search agents. The mathematical model of this phase is formulated as follows:

while $t < \frac{1}{3} * t_{max}$

$$\vec{S}_i = \vec{R}_B \otimes (\vec{E}_i - \vec{R}_B \otimes \vec{py}_i) \tag{2}$$

$$\vec{py}_i = \vec{py}_i + (P * \vec{R} \otimes \vec{S}_i) \tag{3}$$

$\vec{S}_i$ is a vector used to store the step sizes added to the current position to generate a new one. $\vec{R}_B$ indicated the Brownian motion and generated randomly using the Gaussian distribution, $\otimes$ is the element by element multiplication, P is a fixed-value and recommended by the author 0.5, $\vec{R}$ is a vector having a random number generated uniformly between a value greater than or equal to 0 and less than 1. t is the current generation, and t_{max} is the maximum function evaluations.

After a specific number of iterations, specifically $\frac{1}{3}t_{max}$, the exploration phase is about to be terminated, while the exploitation operator using the levy strategy is about to be started. Based on that, this phase will be mixed between exploration and exploitation capabilities. In this phase, the authors divide the population into two halves: the first one will explore using the Brownian strategy, and the second one will exploit using the levy strategy. Generally, this phase is mathematically modeled using the following formula:

While $\frac{1}{3} * t_{max} < t < \frac{2}{3} * t_{max}$

- For the first half:

$$\vec{S}_i = \vec{R}_L \otimes (\vec{E}_i - \vec{R}_L \otimes \vec{py}_i) \tag{4}$$

$$\vec{py}_i = \vec{py}_i + (P * \vec{R} \otimes \vec{S}_i) \tag{5}$$

- For the second half:

$$\vec{S}_i = \vec{R}_B \otimes (\vec{R}_B \otimes \vec{E}_i - \vec{py}_i) \tag{6}$$

$$\vec{py}_i = \vec{E}_i + P * CF \otimes \vec{S}_i \tag{7}$$

$\vec{R}_L$ indicates a vector involving numerical values generated randomly using the levy flight. CF is a variable related to the iteration to minimize the step size by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

increasing the current iteration. CF is mathematically built using the following equation:

$$CF = (1 - \frac{t}{t_{max}})^{2\frac{t}{t_{max}}} \tag{8}$$

After terminating the previous phase that relates the exploration and exploitation capabilities with each simultaneously, the exploitation capability will completely dominate the optimization process within the rest of the iteration. this phase is mathematically formulated according to the following formula:

while $it > \frac{2}{3} * max_{iter}$

$$\vec{S}_i = \vec{R}_L \otimes (\vec{R}_L \otimes \vec{E}_i - \vec{py}_i) \tag{9}$$

$$\vec{py}_i = \vec{E}_i + P * CF \otimes \vec{S}_i \tag{10}$$

In addition, due to the eddy information or the fish aggregating devices (FADs), the predators spend 80% of their search time in the surrounding environment, while the rest time goes away into another position in the search space to find another environment full of the prey and mathematically modeled as:

$$\vec{py}_i = \begin{cases} \vec{py}_i + CF[\vec{X}_L + R \otimes (\vec{X}_U - \vec{X}_L)] \otimes \vec{U} \\ \quad if \qquad\qquad r < FADs \\ \vec{py}_i + [FADS(1-r) + r](\vec{py}_{r1} - \vec{py}_{r2}) \\ \quad if \qquad\qquad r > FADs \end{cases} \tag{11}$$

r a random number randomly assigned a value greater than or equal to 0 and less than 1. $FADs = 0.2$ is used to show how far the FADs will impact the optimization process. $\vec{U}$ is binary vector generated based on Eq. 12. $\vec{py}_{r1}$,and $\vec{py}_{r2}$ are two random predators from the population.

$$\vec{U} = \begin{cases} 1 & if \quad \vec{r_1} < FADs \\ 0 & if \quad \vec{r_1} > FADs \end{cases} \tag{12}$$

Finally, the predators will save the best solution for each $\vec{py}_i$ in the previous generation, then compare this best with the updated one, and the best one out of them will be used in the next generation; this process is known as the memory saving.

III. COMPUTATION MODEL

Assuming that there is a job had by a user and this job consist of several tasks $t_i = 1, 2, 3, \ldots, d|d$ indicates the number of tasks; each task has an associated workload named w_i with a unit of million instructions (MI). Also, there is n fog virtual machines (VMs), where each one could process only a task t_i at time. The TSFC seeks to find the best assignments of tasks to those VMs, which minimizes the execution time required for those tasks to improve the quality of services presented to the users, especially in critical applications. To measure the quality of services which results of the obtained assignment, four performance metrics: makespan, flow time, energy, and carbon dioxide emission rate will be used. Each of which will be elaborated. The makespan and carbon dioxide emission rate consider contradictory objectives, where improvement of one may produce with deteriorating the others. For example, the makespan shows the maximum execution time required, while energy indicates the total energy consumed by all VMs to complete the execution of the tasks, and the carbon dioxide emission rate will subsequently increase due to increasing the energy. The VMs could be in two states either idle or active state and in the idle state, the energy consumed is about 60% of the active state as mentioned in [25]. Therefore, when the makespan is high and the execution time of a VM in the active state is low, this will result in idle this VM for a long time, and subsequently, the energy consumed will be significantly increased. Also, some jobs within the fog systems may wait for a long time, so the flow time will increase although minimizing the makespan related to the throughput. Hence, the relation between makespan and the three others is contradictory. Therefore, the tasks must be distributed in a better order between VMs to involve the equilibrium among various objectives. All of those will be discussed in the rest of this section.

A. Makespan.

At the start time, each VM_j will be assigned a time of 0. Afterward, each one will be assigned a set of tasks to execute them. The execution time Et_j for each task on the virtual machine VM_j will be calculated and added with each other until the VM finishes performing all the tasks assigned to it. The total Et_j needed by the VM_j until completing the execution of all the tasks assigned into them will be calculated and the highest total execution time needed by a VM to complete its mission will be considered as the makespan.

B. Energy.

Each virtual machine may be in one of two states: active and idle. At the idle state, the energy consumed by each VM is considered 60% of the energy consumed in the active state. The total energy consumed by each VM is a summation of the energy consumed in the active and idle states. The total consumed energy by each virtual machine can be mathematically calculated according to the following formula:

$$E(VM_j) = (Et_j \times b_j + (MK - Et_j) * a_j) \times s_j \tag{13}$$

$$b_j = 10^{-8} \times s_j^2 \tag{14}$$

$$a_j = 0.6 * b_j \tag{15}$$

Where b_j is the energy consumed in the active state of each VM. s_j indicates the processing speed of the j^{th} VM (million instructions per second) [52]. The energy consumed by all the virtual machines is a summation of the total energy consumed by each one, as shown in

the following equation.

$$total\_energy = \sum_{j=1}^{n} E(VM_j) \tag{16}$$

C. Carbon dioxide emission rate.

To compute the carbon dioxide emitted by each VM, we used the following formula:

$$carbon\_emission\_rate = \sum_{k=1}^{4} total\_energy \times sh_k \times emission\_factor_k \times ratio \tag{17}$$

The power is a mix of coal, oil, natural gas, and non-fossil products numbered as $k = 1, 2, 3, 4$, respectively. sh_k refers to the share of k^{th} energy source in the total energy consumption. Moreover, $emission_factor_k$ expresses the k^{th} energy source's emission factor and is $0.5825, 0.7476, 0.4435, 0$ for oil, coal, natural gas, and non-fossil products, respectively [53]. The ratio is $44/12$ and indicates the converting ratio from carbon to carbon dioxide [54].

D. Flow Time (f_t).

This metric is used to compute the time consumed by a task within the fog system. The total time consumed by all tasks within their VMs are called flow time and computed using the flowing formula:

$$f_t = \sum_{i=0}^{n-1} f_i \tag{18}$$

f_i indicates the flow time of all tasks assigned to the i^{th} VM.

E. Normalization and scaling phase.

Because the MPA's obtained solution is continuous, and the MTSFC is discrete, the normalization and scaling strategy is used to convert them into discrete values. The obtained constant values will be normalized between 0 and 1 and scaled in [1, n] using the following formula:

$$NS_i(t+1) = \frac{py_i(t+1) - L}{U - L} * ((n-1)+1) \tag{19}$$

L is the smallest value among the obtained continuous values for the solution $py_i(t+1)$, U the maximum value in the same solution $py_i(t+1)$, respectively. $py_i(t+1)$ is the value in each dimension in the updated solution. $NS_i(t+1)$ is the scaled normalized value of i^{th} dimension.

IV. THE PROPOSED MULTIOBJECTIVE TASK OFFLOADING APPROACH: MIMPA.

This part will illustrate our methodology used to solve the multi-objective task scheduling problem for fog computing (MTSFC). The makespan indicates the highest total execution time needed by a VM to complete all tasks assigned, while consumed energy and carbon dioxide emission rate are directly proportional contradicted to the makespan. Therefore, The tradeoff between minimizing the makespan and carbon dioxide emission rate is tackled in this paper by using the Pareto optimality to find a solution that couldnŠt be improved by any of its objectives without degrading the other; this solution is known as a Non-dominated solution. In addition to that, we used an archive to store all the dominant solutions obtained so-far. Furthermore, we improve the levy flight strategy to increase its capability to find better solutions. Generally, this section is organized as follows:

1) Section IV-A Explains the initialization phase.
2) Section IV-B describes the Pareto optimality.
3) Section IV-C shows our improvement on the levy Flight.
4) Section IV-D shows the Modified Polynomial mutation mechanism.

A. INITIALIZATION.

In the start, suppose we have a system consist of n virtual machines (VM) and a number of d tasks will be assigned to those VM in the order that will minimize the makespan and the carbon dioxide emission rate. To solve that using an optimization algorithm, we will create N solutions, where each one consists of d tasks (dimensions or decision variables) and initialized randomly at the range of 0 and (n-1) using the following formula:

$$\vec{X} = \vec{r}.n \tag{20}$$

$\vec{r}$ is a vector involving numerical values generated randomly in a range greater than or equal to 0 and less than 1. After initializing each solution, we will measure the dominance of this solution compared to the other, as shown in the following section.

B. PARETO OPTIMALITY.

In the problem with a single-objective, the best solution could be found easily based on comparing the fitness values of the solutions with each other and the one with the smallest fitness, if the problem is minimized, will be used as the best, but if the problem is maximized, then the highest one will be considered as the best one. Unfortunately, if we deal with our problem as a multiobjective one, we don't know how to determine the best solution because more than one objective needs to be optimized. Therefore, in the multiobjective problem, we need to find a solution that will optimize all the objectives together. If the solution obtained improves one objective and deteriorates the other, it couldn't be considered a solution to our problem. As a result, the Pareto optimality appears to search for the solution that will optimize all the objectives together based on their dominance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

The Pareto optimality said that solution A dominates solution B if and only if solution A could reach a better value for at least an objective while the others are equal; A is a non-dominated solution. So, Pareto optimality could find the best solutions for the multiobjective optimization problem, since it could cover all the possible solutions that achieve the best trade-offs among the objectives [55], [56].

C. LEVY FLIGHT STRATEGY.

The French mathematician Paul Levy proposed the levy flight that was based on occasionally jumping over long distances to increase the diversity in the population and the short distances to accelerate the convergence toward the best-solution. Each levy flight step consists of two control factors: one to determine the direction of the flight and is based on the uniform distribution function. The other is the step size to obey the levy distribution. Mathematically, the step size under the levy flight could be generated as follows:

$$R_L = \alpha \frac{u}{|v|^{\frac{1}{\beta}}} \tag{21}$$

α is a fixed-value used to determine the weight of the levy step. Both u and v is based on Gaussian distribution and formulated as:

$$u \sim N(0, \sigma_u^2), v \sim N(0, \sigma_v^2) \tag{22}$$

Where

$$\sigma_v = 1 \tag{23}$$

$$\sigma_u = \left(\frac{\gamma(1+\beta)\sin(\frac{\pi\beta}{2})}{\gamma((1+\beta)/2)\beta 2^{(\beta-1)/2)}}\right)^{\frac{1}{\beta}} \tag{24}$$

Where β as used in the MPA algorithm, γ is a Gamma function that is expressed as:

$$\gamma(z) = \sqrt{\frac{2\pi}{z}} * \left(\frac{1}{e} * (z + 1/(12 * z - 1/(10 * z)))\right)^z \tag{25}$$

Our idea was based on using the Cauchy distribution instead of the Gaussian distributions to give the algorithm high ability on exploiting around the best so-far solution with giving some exploration capability to help the optimization algorithms, generally, in getting out of the local minima if the best-so-far is so. Mathematically, the Cauchy distribution will be formulated:

$$C(y) = y \tan(\pi(r_1 - 0.5)) \tag{26}$$

$$u = C(y)\sigma_u \tag{27}$$

$$v = C(y) \tag{28}$$

Where y and r_1 are two numbers generated randomly based on the uniform distribution. Generally, the main steps of the multiobjective MPA improved based on generating the levy flight under Cauchy distribution instead of Gaussian is shown in Algorithm 2 and abbreviated as MIMPA for tackling MTSFC under the Pareto optimality to find the non-dominated solutions and an external archive to save the non-dominated solutions obtained within the optimization process.

In Fig. 1, we show the step sizes generated by the levy flight under Gaussian and Cauchy to determine the flight's direction. After inspecting this figure, we found that the levy flight under Cauchy distributions could generate step sizes with different levels to cover the regions around the best so-far solution as much as possible. As shown in the figures based on Cauchy distribution, some states' step sizes are so small to accelerate the convergence as fast as possible. At another time, the step size is medium, which not large to promote the exploration and not short of increasing the exploitation. This value will help the algorithm search inside regions not far away from the best solution, and subsequently, the possibility of finding better is so possible. In some short times, the step sizes generated under Cauchy are relatively large to help the algorithm in voiding stuck into local minima that may fall into if the best-so solution is so.

On the contrary, the step sizes generated under Gaussian are un-regular and limited to only promoting the exploitation capability. Subsequently, if the best-so-far solution is local minima, then the possibility of getting out of it is impossible. Suppose we observe the following figures based on running the algorithm under the Cauchy and Gaussian three independent times. In that case, we will find that the values generated under the Cauchy three times are near to each other with some small difference to increase one of the operators: exploration and exploitation each time against the other. On the contrary, each run's step size values are significantly different from each other and move in one direction, as shown in Fig. 1f. This may take the solution in some states into the local minima problem.

It is worth saying that some researchers have compared the difference between the levy, normal, and Cauchy distributions, but no one tries to generate randomly the value of u and v using the Cauchy distribution instead of the normal distribution to increase the search area explored by the levy flight [57].

D. MODIFIED POLYNOMIAL MUTATION MECHANISM.

Deb [58] proposed a slight modification to the standard polynomial mutation for overcoming the premature convergence, which is caused as a result of arriving the variable to its boundary, and subsequently the diversity of the population required to avoid falling into local minima is significantly reduced. Generally, the mathematical model of the modified polynomial mutation strategy according to [58] is as that:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

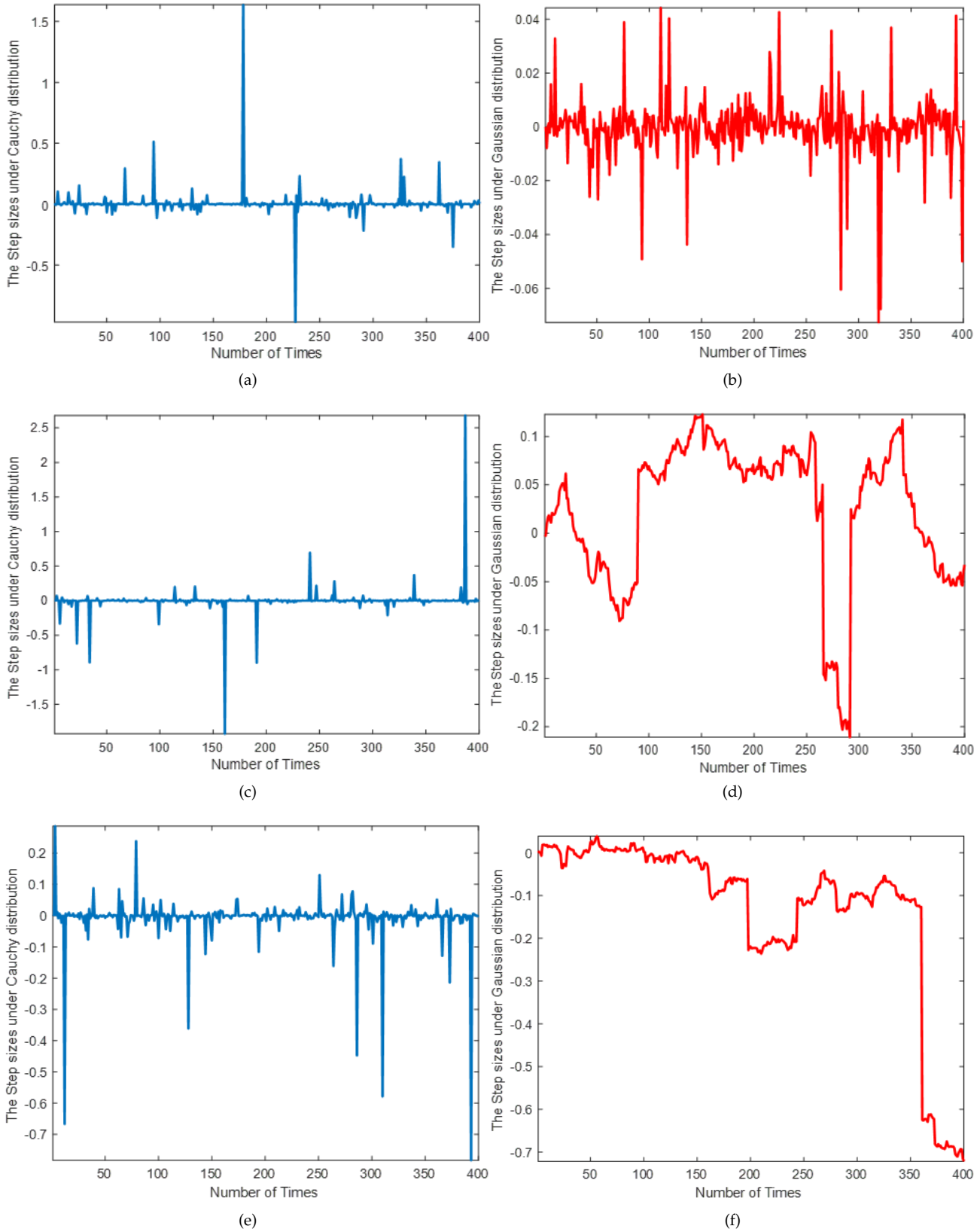


FIGURE 1: Comparison between Levy Flight under Gaussian and Cauchy distributions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

$$\delta_1 = \frac{py_{ij} - \vec{X}_{Lj}}{X_{Uj} - \vec{X}_{Lj}} \tag{29}$$

$$\delta_2 = \frac{X_{Uj} - py_{ij}}{X_{Uj} - X_{Lj}} \tag{30}$$

$$\delta = \begin{cases} [2r + (1-2r)(1-\vec{\delta}_1)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} \\ ,\vec{r} \leqslant 0.5 \\ 1 - [2(1-r) + 2(r-0.5)(1-\delta_2)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} \\ ,otherwise \end{cases} \tag{31}$$

Afterward, the mutated solution will be generated according to the following formula based on a certain mutation probability p_m:

$$py_{ij} = py_{ij} + \delta * (X_{Uj} - X_{Lj}) \tag{32}$$

Where py_{ij} indicates the j^{th} dimension of the i^{th} prey, and X_{Uj} and X_{Lj} are the upper and lower bound values of the same dimension. r and η_m express a random number generated between 0 and 1, and the index for the polynomial mutation, respectively. At the start of the optimization process, to increase the exploration capability of the algorithms, the mutation probability p_m will start with a relatively large value, which gradually decreases with the iteration to increase the exploitation operator in the hope of finding a better solution. Mathematically, p_m is updated using the following formula:

$$p_m = \tau * \frac{t_{max} - t}{t_{max}} \tag{33}$$

Where τ is a predefined probability value at the range of 0 and 1. In our experiments, the value of τ is set to 0.2. In algorithm 1, the main steps of polynomial mutation are presented.

Algorithm 1 polynomial mutation (PM).

```
Input: the current prey, py
Update p_m using eq. 33
for i=0 to d do
    r_2: generate a random number between 0 and 1
    if r_2 < p_m then
        Mutate the variable py_i using eq. 32.
    end if
end for
Output: return py
```

After that, algorithm 1 is integrated with algorithm 2 after Line 21 and Line 26 to mutate each prey in the population as an attempt to avoid stuck into local minima and promote the exploration capability for reaching better outcomes; this version is abbreviated as MHIMPA. Also, the effect of integrating the polynomial mutation with the standard multi-objective MPA (MMPA) is investigated in a version abbreviated as MHMPA.

Algorithm 2 MIMPA

```
1: Initialization Step, (P, py_i)
2: archive: an empty pool to contains the non-dominated solution within the optimization process
3: while t < t_max do
4:    Apply Normalization and scaling phase to p⃗y_i
5:    Add p⃗y_i to the archive if it is non-dominated.
6:    Construct E based on a solution selected randomly from archive.
7:    Achieve the memory saving.
8:    Construct E.
9:    Update CF based on Eq. 8
10:   for ∀py_i do
11:      if t < (1/3)t_max then
12:         Reposition p⃗y_i based on Eq. 3
13:         if 1/3 * t_max < t < 2/3 * t_max then
14:            if i < 1/2 * N then
15:               update p⃗y_i based on Eq. 5
16:            else
17:               reposition p⃗y_i according to Eq. 7
18:            end if
19:         end if
20:      else
21:         reposition p⃗y_i according to Eq. 10
22:      end if
23:   end for
24:   t=t+N.
25:   Apply Normalization and scaling phase to p⃗y_i
26:   Add p⃗y_i to the archive if it is non-dominated.
27:   Achieve the memory saving
28:   update E if there is a predator better
29:   Achieve the FADS for ∀p⃗y_i using Eq. 11
30:   t=t+N.
31: end while
```

V. RESULTS AND DISCUSSION.

During displaying in this section, we will show the efficacy of the proposed algorithm when tackling the MTSFC under four metrics: Make-span, Flow Time, dioxide emission rate, and energy consumed, in addition to comparing with a number of well-known robust multiobjective optimization algorithms described as follows:

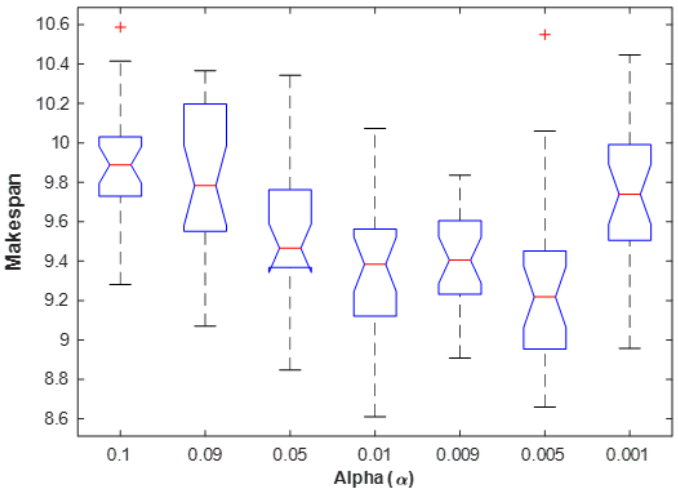
- Multiobjective Marine predators algorithm (MMPA) [51].
- A non-dominated sorting genetic algorithm II (NSGAII) [59].
- NSGAIII [60].
- Speed-constrained Multiobjective particle swarm optimization (SMPSO) [61].
- Particle swarm optimization-based multiobjective optimization using crowding, mutation, and ϵ-dominance (OMOPSO) [62].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

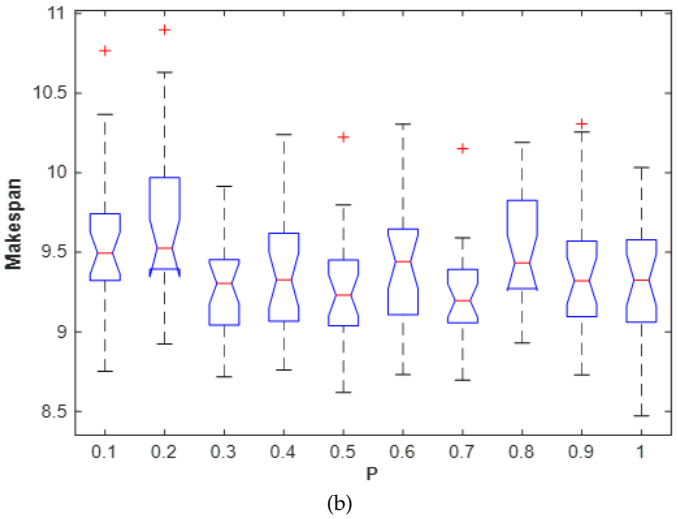
- Multiobjective PSO based on decomposition [63].
- Unary diversity indicator based on reference vectors (DIR) based NSGAII (DNSGAII) [64].
- Modified Indicator based Evolutionary Algorithm (MIBEA) [65].

Regarding those algorithms parameter settings, we used the same parameter values cited in the published paper and shown in Table 2. It is worth mentioning that all the algorithms are running 25 independent times with $t_{max} = 50000$ and $N = 100$ to make a fair comparison using the JMetal framework [66] on a device with the following capabilities: $IntelCorei7-4700MQCPU@2.40GHz, 16GB$ of memory, and equipped with Windows 10 platform.

In Fig. 2, different P and α values are observed to see the best values for them. Specifically, the best values for P and α based on the average makespan shown in this figure are 0.7 and 0.005, respectively.



(a)



(b)

FIGURE 2: Observation of different values for P and α.

Finally, the rest of this section is organized as follows:

1) Section V-A: Dataset Descriptions.
2) Section V-B: Comparison under the first dataset.
3) Section V-C: Comparison under the second dataset.

A. DATASET DESCRIPTIONS.

Through our simulation-based experiments, the datasets are divided into two groups: the first one contains different tasks lengths of 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1200, and 1400 with a fixed VM length equal to 50, and the second one consists of a constant task length equal to 500 and different VM lengths of 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 150. All the first group tasks are heterogeneous and have workloads generated randomly between 1000 and 10000. Each VM out of 50 ones has a processing speed of 2000 for the first half and 4000 MIPS for the second half. Regarding the cost related to each VM out of 50 ones, the first half of 50 VMs has a cost of 200$ while the second half has a cost of 400$; the difference in costs between two halves is related proportionally with the processing speed.

For the second group, Each VM length is divided into halves: the first one will have a processing speed of 2000 MIPS with a cost of 200$, and the second half has a processing speed of 4000 with a cost of 400$.

B. COMPARISON OF DIFFERENT TASK SIZES.

After running each algorithm 25 independent runs and calculating the average of the obtained make-span values in the archive within those runs on each task size (TS), we show those averages in Table 3. Inspecting this table show that our proposed algorithm (MIMPA) could reach the lowest make-span for all the task sizes in comparison to MMPA, in addition to overcoming the other algorithms except MHMPA and MHIMPA for most of the task sizes. Furthermore, the average of the makespan values obtained by each algorithm within those runs is shown in the last row of this table. From those averages, it is clear that MHMPA could reach the lowest average makespan with a value of 29.120, MHIMPA comes after MHIMPA with a value of 30.90, and MIMPA occupies the third rank after H-MIMPA with an amount of 33.04 while NSGAIII comes as the last one with a value of 43.46. Based on this analysis, our improvement based on using the Cauchy distribution with the Levy flight mechanism instead of the Gaussian distribution could significantly affect the performance of the MIMPA for the various task sizes. This is due to the ability of the Cauchy distribution is giving MIMPA a high ability on exploiting with a small ratio of the exploration around the best-so-far solution that enables the algorithm of avoiding stuck into local minima. But unfortunately, integrating the PM with the MIMPA affect negatively its performance contradicted the MMPA that could come true with the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

TABLE 2: Parameter settings for the Algorithms

Algorithm	Parameter	Value	Algorithm	Parameter	Value
NSGAII, DNSGAII	Selection	binary tournament	MMPA/ MHMPA	P	0.7
	Recombination	pc = 0.9		τ	0.2
	Mutation	Polynomial pm = 1.0/d		Archive size	100
	η_m	20		FADs	0.2
SMPSO	Selection	binary tournament	NSGAIII	η_m	5
	Mutation	Polynomial pm = 1.0/d		Recombination	pc = 0.9
	Archive size	100		Mutation	Polynomial pm = 1.0/n
OMOPSO	Selection	binary tournament	DMOPSO	Archive size	100
	Mutation	uniform pm = 1.0/d		maxAge	2
				r1Min, r2Min	0
		Nonuniform pm=1.0/d for iterations=250		r2Min, r2max	1.0
				c1Min, c2Min	1.5
				c2Min, c2Max	2.5
				Min and max weight	0.1, and 0.4
MIBEA	Selection	binary tournament			
	Recombination	pc = 0.9			
	Mutation	Polynomial pm = 1.0/d			
	Archive size	100			

lowest makespan with the PM. As a result, MHMPA could reach a better assignment of the tasks to the VMs that alleviates the maximum execution time.

We calculated the average of the carbon emission rate generated by each VM on each task size and introduced it in Table. 4. This table shows that our proposed algorithm, MHMPA, could reach less carbon emission rate compared with the other algorithms for the task sizes greater than and equal to 300, while NSGAII could be the best for the tasks sizes smaller than that. Afterward, the average of each column in this table is calculated and shown in its last row from which it can be concluded that the proposed algorithm is a friend to the environment due to the reduction of the amount of the carbon emission rate transmitted by each VM.

Regarding the flowtime, Table 5 shows the average flowtime obtained by each algorithm on each task size. According to this table, our proposed algorithm: MHMPA could get less flowtime except for the task size 200, where NSGAII could be the best. The average of each column in Table 5 is calculated and displayed in its last row, which shows the superiority of MHMPA with a value of 12550 compared to the others.

Table 6 shows the energy consumed by each VM according to the assignment achieved by each algorithm on them. This table shows that our proposed algorithm is considered the one with less energy consumed. To confirm our analysis graphically, we calculate the average of the make-span obtained by each algorithm on all the TSs. According to this analysis, our proposed algorithm could reach the best assignment to the tasks on the VMs that will reduce the energy consumed and this shows that our proposed is considered an energy saver.

Fig. 3, 4 draws the boxplot of the makespan values obtained by the different algorithms on the various task sizes. Of this figure, it is obvious that the proposed algorithms: MHMPA and MHIMPA are competitive with the DNSGAII and NSGAII for the task sizes smaller than 500 and significantly superior for the other sizes. In comparison to the other algorithms under the different task sizes, MHMPA is better in terms of the minimum, maximum, median, first quartile, and third quartile. Also, of this figure, the MIMPA could be superior to the standard one: MMPA in terms of the minimum, median, first quartile, and third quartile in most cases.

In Table 7, the proposed algorithm (MHMPA) is compared with the others using the Wilcoxon rank-sum test [67] with a confidence level of 5%. This test is based on two hypotheses: Null hypothesis and alternative hypothesis. Null hypothesis indicates that there is no difference between two compared algorithms; in this case, p presented in Table 12 includes a value higher than the confidence level. Meanwhile, the alternative assumes that there is a difference between the outcomes obtained by a pair of algorithms; this hypothesis is achieved when p includes a value smaller than the confidence level. Table 7 shows the outcomes of comparing MHMPA with the other compared algorithms under this test. According to this table, the alternative hypothesis is come true for most cases and this elaborates the difference between the findings of the proposed and the other algorithms.

C. COMPARISON UNDER DIFFERENT VM LENGTHS.

In this section, we will investigate our proposed algorithm's performance when the VMs length is ranging between 10 and 150. Each algorithm is executed 25 independent times, then the average of the make-span obtained by each one is introduced in Table 8. Based on this table, MMMPA could come true less make-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

TABLE 3: Average Make-span obtained by MHMP, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII and MIBEA.

TS	MHMPA	MHIMPA	MIMPA	MMPA	NSGAII	SMPSO	OMOPSO	DMOPSO	NSGAIII	DNSGAII	MIBEA
T100	5.0278	5.1896	5.6108	6.1074	**4.9878**	6.6641	6.6205	6.9746	6.6266	5.0235	5.8934
T200	**9.1845**	9.4237	9.9986	10.6824	**9.1365**	12.3635	12.0578	12.2259	12.8104	9.1876	10.6612
T300	**13.5032**	13.9366	14.8210	15.5762	14.3920	18.2765	17.7319	17.6025	18.8347	13.8217	16.1397
T400	**17.4913**	18.0919	19.2752	20.1996	18.2235	24.0523	23.0641	21.7606	25.4765	18.7492	21.6907
T500	**20.9124**	21.9547	23.1915	24.5417	22.4652	28.9338	28.5551	26.0999	30.8837	23.5065	26.3205
T600	**25.6333**	26.9498	28.3062	30.4910	28.1744	36.4110	34.8252	31.8947	38.5582	28.6224	32.5196
T700	**29.5072**	30.9683	32.9315	35.4979	33.1232	41.4829	40.3611	36.3815	44.2462	33.9607	38.2895
T800	**34.2028**	36.1143	38.4179	40.3963	39.6958	47.4648	46.5753	40.7703	50.9968	38.7626	43.6892
T900	**37.2800**	39.2066	42.2418	45.5176	44.2421	52.4935	50.8419	44.9733	56.5694	42.9891	49.1916
T1000	**42.4600**	45.3499	48.2176	50.7265	50.4178	59.7854	57.8889	50.3909	64.0320	49.7062	55.0514
T1200	**50.2280**	53.3478	57.7309	61.9179	60.1966	71.2108	69.2458	59.0504	75.7152	60.2278	65.1412
T1500	**64.0123**	70.2980	75.7352	83.0753	77.6102	90.3850	87.6848	75.6600	96.7696	78.4363	84.5753
Avg	**29.12023**	30.9026	33.03985	35.39415	33.55543	40.79363	39.62103	35.31538	43.45994	33.5828	37.43028

Bold value indicates the best value

TABLE 4: Average carbon dioxide emission rate obtained from MHMPA, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII, and MIBEA.

TS	MHMPA	MHIMPA	MIMPA	MMPA	NSGAII	SMPSO	OMOPSO	DMOPSO	NSGAIII	DNSGAII	MIBEA
T100	1.867E+05	1.960E+05	2.068E+05	2.202E+05	**1.853E+05**	2.249E+05	2.243E+05	2.362E+05	2.220E+05	1.864E+05	2.029E+05
T200	3.505E+05	3.666E+05	3.812E+05	3.983E+05	**3.474E+05**	4.234E+05	4.146E+05	4.282E+05	4.315E+05	3.484E+05	3.767E+05
T300	**5.233E+05**	5.475E+05	5.716E+05	5.907E+05	5.370E+05	6.293E+05	6.126E+05	6.242E+05	6.431E+05	5.255E+05	5.710E+05
T400	**6.850E+05**	7.195E+05	7.450E+05	7.689E+05	6.932E+05	8.288E+05	7.993E+05	7.914E+05	8.618E+05	7.042E+05	7.612E+05
T500	**8.280E+05**	8.697E+05	9.022E+05	9.340E+05	8.502E+05	1.000E+06	9.832E+05	9.555E+05	1.047E+06	8.706E+05	9.265E+05
T600	**1.022E+06**	1.077E+06	1.100E+06	1.154E+06	1.060E+06	1.251E+06	1.208E+06	1.175E+06	1.303E+06	1.070E+06	1.147E+06
T700	**1.174E+06**	1.233E+06	1.284E+06	1.338E+06	1.235E+06	1.429E+06	1.391E+06	1.341E+06	1.497E+06	1.252E+06	1.339E+06
T800	**1.362E+06**	1.435E+06	1.485E+06	1.535E+06	1.459E+06	1.640E+06	1.610E+06	1.519E+06	1.729E+06	1.439E+06	1.537E+06
T900	**1.498E+06**	1.576E+06	1.636E+06	1.707E+06	1.619E+06	1.814E+06	1.763E+06	1.686E+06	1.913E+06	1.593E+06	1.719E+06
T1000	**1.701E+06**	1.798E+06	1.869E+06	1.913E+06	1.842E+06	2.062E+06	2.004E+06	1.897E+06	2.164E+06	1.826E+06	1.933E+06
T1200	**2.015E+06**	2.113E+06	2.213E+06	2.311E+06	2.192E+06	2.449E+06	2.392E+06	2.222E+06	2.559E+06	2.192E+06	2.289E+06
T1500	**2.579E+06**	2.762E+06	2.850E+06	3.013E+06	2.822E+06	3.119E+06	3.041E+06	2.850E+06	3.274E+06	2.839E+06	2.963E+06
Avg	**1.16E+06**	1.22E+06	1.27E+06	1.32E+06	1.24E+06	1.41E+06	1.37E+06	1.31E+06	1.47E+06	1.24E+06	1.31E+06

Bold value indicates the best value

TABLE 5: Average Flow Time obtained from MHMPA, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII, and MIBEA.

TS	MHMPA	MHIMPA	MIMPA	MMPA	NSGAII	SMPSO	OMOPSO	DMOPSO	NSGAIII	DNSGAII	MIBEA
T100	296	296	301	303	**295**	327	326	333	336	297	325
T200	937	950	960	981	**936**	1031	1034	1023	1059	940	1028
T300	**1972**	2022	2073	2083	2050	2193	2215	2159	2208	2024	2184
T400	**3446**	3549	3587	3637	3541	3829	3837	3677	3914	3582	3866
T500	**5079**	5252	5329	5415	5283	5604	5747	5434	5759	5358	5700
T600	**7310**	7557	7602	7784	7641	8140	8161	7843	8315	7660	8127
T700	**9679**	9932	10167	10344	10128	10698	10897	10315	10931	10274	10839
T800	**12865**	13263	13468	13622	13632	14164	14374	13564	14463	13514	14253
T900	**15673**	16196	16458	16776	16787	17239	17448	16455	17790	16553	17571
T1000	**20227**	20901	21417	21445	21422	22267	22508	21076	22874	21448	22429
T1200	**28253**	28980	29651	30095	30225	31167	31478	29744	31940	30266	31186
T1500	**44859**	46664	47043	48306	47770	49418	49838	47403	50472	48088	49777
Avg	**12550**	12964	13171	13399	13309	13840	13989	13252	14172	13334	13940

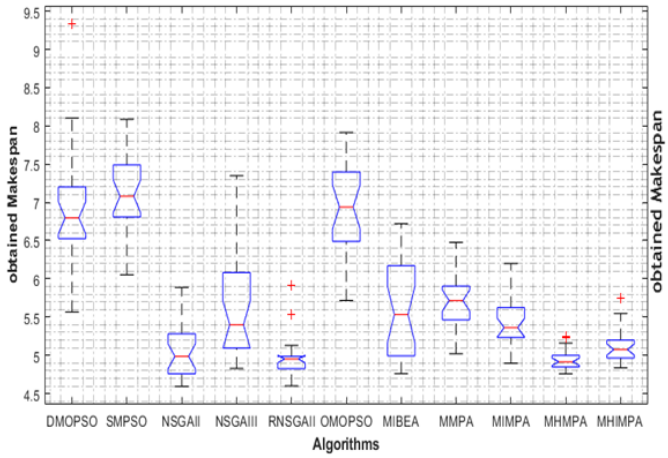
Bold value indicates the best value

TABLE 6: Average energy obtained by MHMPA, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII and MIBEA.

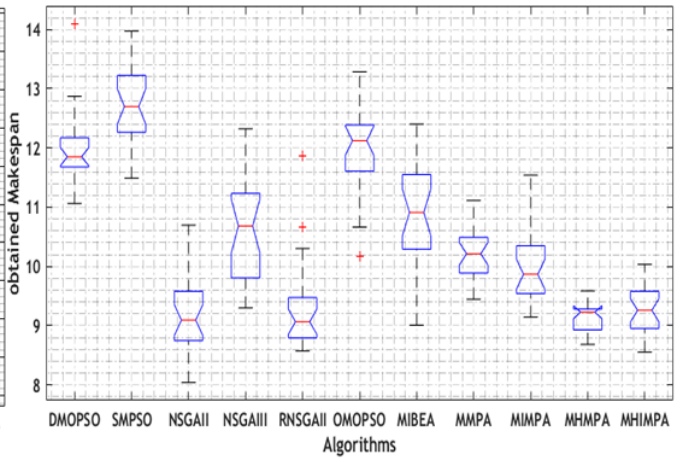
TS	MHMPA	MHIMPA	MIMPA	MMPA	NSGAII	SMPSO	OMOPSO	DMOPSO	NSGAIII	DNSGAII	MIBEA
T100	**7.932E+04**	8.326E+04	8.788E+04	8.801E+04	7.875E+04	9.557E+04	9.530E+04	1.004E+05	9.432E+04	7.922E+04	8.621E+04
T200	**1.489E+05**	1.557E+05	1.620E+05	1.575E+05	1.476E+05	1.799E+05	1.762E+05	1.820E+05	1.833E+05	1.480E+05	1.601E+05
T300	**2.223E+05**	2.326E+05	2.429E+05	2.335E+05	2.282E+05	2.674E+05	2.603E+05	2.652E+05	2.732E+05	2.233E+05	2.426E+05
T400	**2.911E+05**	3.057E+05	3.165E+05	3.063E+05	2.945E+05	3.521E+05	3.396E+05	3.363E+05	3.662E+05	2.992E+05	3.234E+05
T500	**3.518E+05**	3.695E+05	3.833E+05	3.687E+05	3.613E+05	4.249E+05	4.177E+05	4.060E+05	4.448E+05	3.699E+05	3.937E+05
T600	**4.341E+05**	4.577E+05	4.673E+05	4.522E+05	4.505E+05	5.315E+05	5.131E+05	4.992E+05	5.535E+05	4.548E+05	4.871E+05
T700	**4.987E+05**	5.240E+05	5.455E+05	5.211E+05	5.249E+05	6.073E+05	5.912E+05	5.700E+05	6.362E+05	5.320E+05	5.691E+05
T800	**5.787E+05**	6.098E+05	6.309E+05	6.025E+05	6.198E+05	6.967E+05	6.840E+05	6.455E+05	7.348E+05	6.115E+05	6.532E+05
T900	**6.367E+05**	6.697E+05	6.952E+05	6.681E+05	6.877E+05	7.707E+05	7.493E+05	7.162E+05	8.130E+05	6.767E+05	7.302E+05
T1000	**7.229E+05**	7.639E+05	7.940E+05	7.526E+05	7.825E+05	8.762E+05	8.515E+05	8.060E+05	9.196E+05	7.760E+05	8.214E+05
T1200	**8.562E+05**	8.978E+05	9.405E+05	8.849E+05	9.313E+05	1.040E+06	1.016E+06	9.441E+05	1.087E+06	9.312E+05	9.727E+05
T1500	**1.096E+06**	1.174E+06	1.211E+06	1.141E+06	1.199E+06	1.325E+06	1.292E+06	1.211E+06	1.391E+06	1.206E+06	1.259E+06
Avg	**4.93E+05**	5.20E+05	5.40E+05	5.15E+05	5.26E+05	5.97E+05	5.82E+05	5.57E+05	6.25E+05	5.26E+05	5.58E+05

Bold value indicates the best value

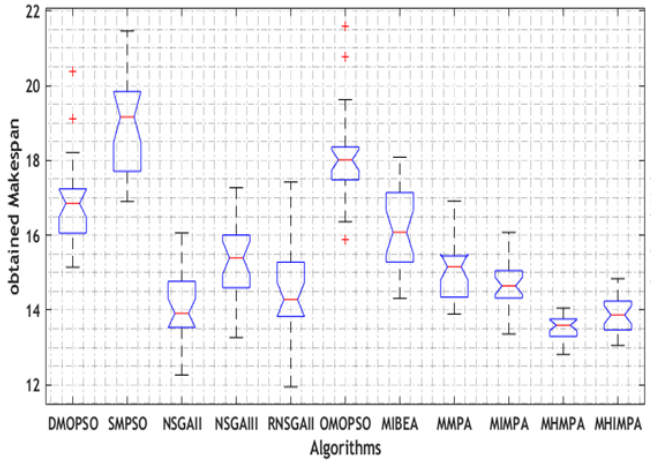
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access



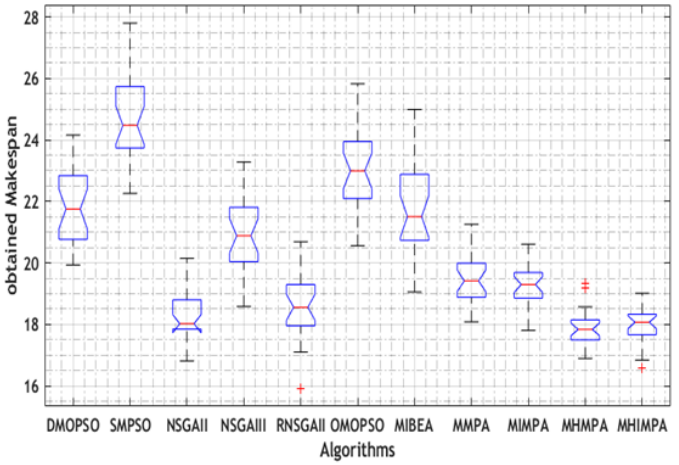
(a) TS=100



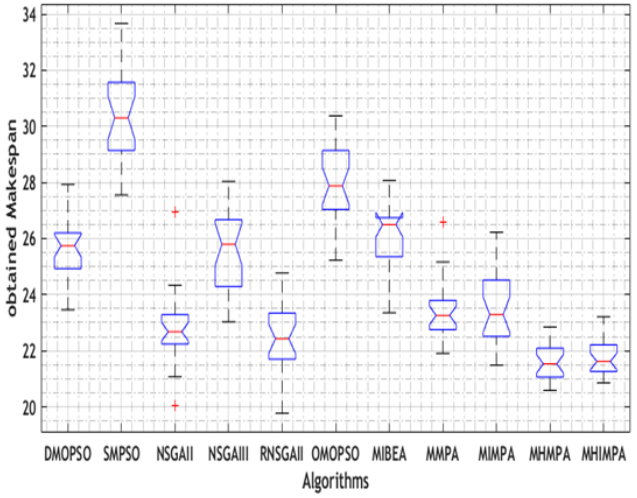
(b) TS=200



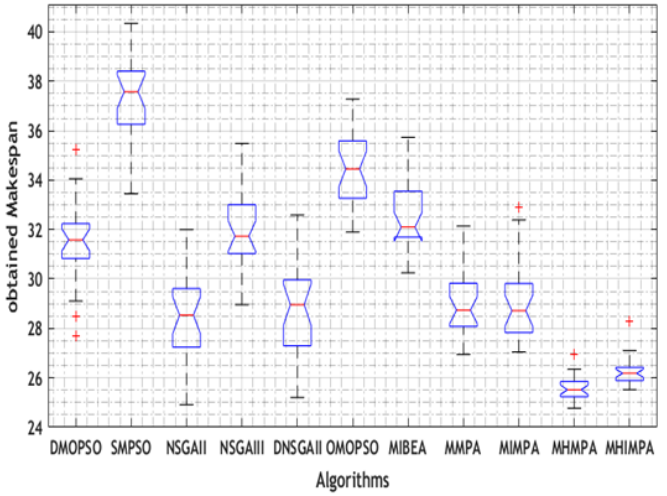
(c) TS=300



(d) TS=400



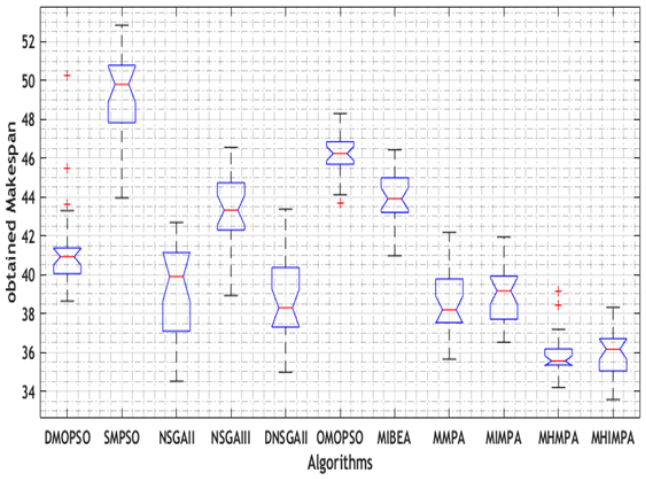
(e) TS=500



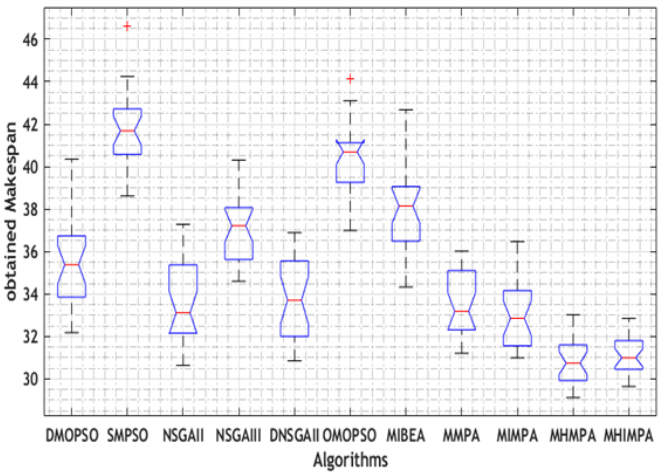
(f) TS=600

FIGURE 3: Comparison among the algorithms based on the boxplot for the makespan values of the various task sizes (100-600).

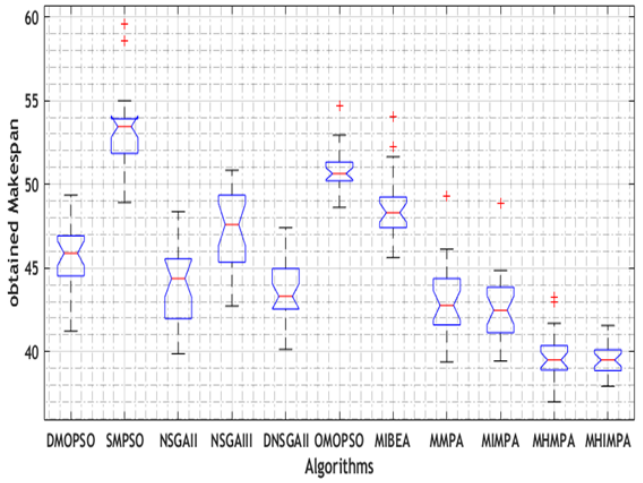
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access



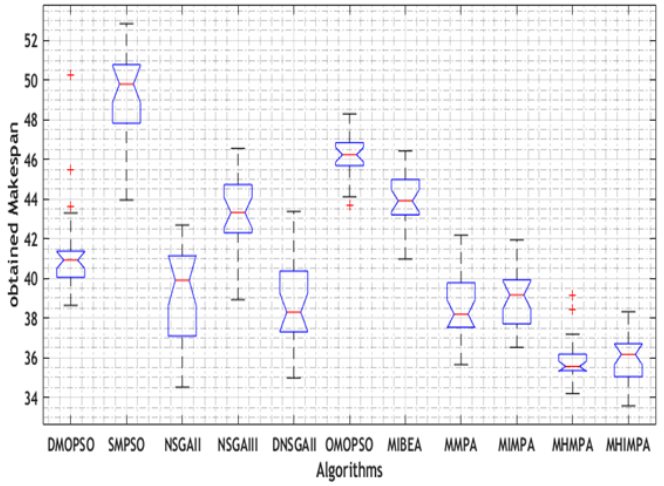
(a) TS=700



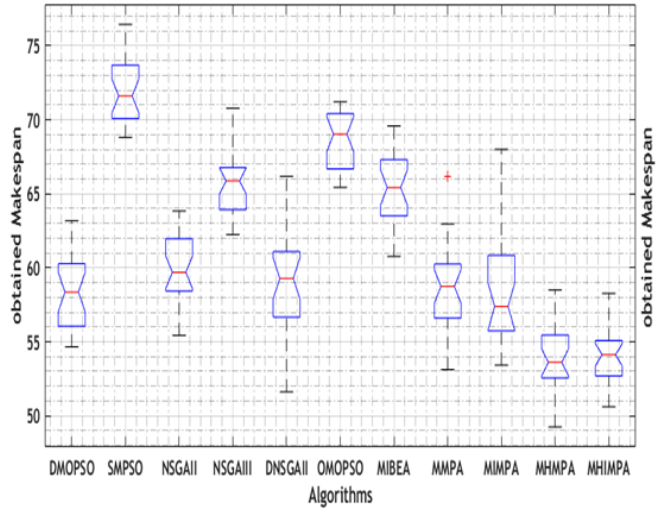
(b) TS=800



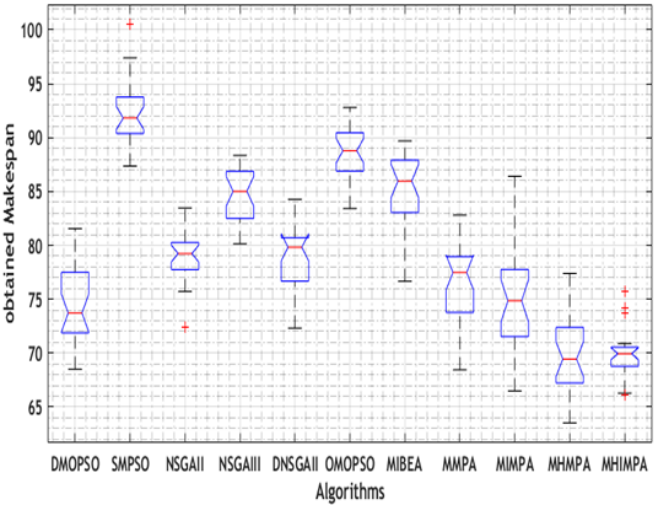
(c) TS=900



(d) TS=1000



(e) TS=1200



(f) TS=1500

FIGURE 4: Comparison among the algorithms based on the boxplot for the makespan values of the various task sizes (700-1500).

TABLE 7: Wilcoxon Rank sum test between MHMPA, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII, and MIBEA under the makespan.

TS		DMOPSO	SMPSO	NSGAII	NSGAIII	DNSGAII	OMOPSO	MIBEA	MMPA	MIMPA	MHIMPA
T100	h-	1.384E-09	1.384E-09	3.715E-01	3.101E-05	9.845E-01	1.383E-09	1.417E-04	2.833E-09	1.765E-07	8.077E-04
	p-	1	1	0	1	0	1	1	1	1	1
T200	h-	1.416E-09	1.414E-09	7.710E-01	3.669E-09	5.220E-01	1.416E-09	1.465E-08	2.898E-09	3.017E-07	1.936E-01
	p-	1	1	0	1	0	1	1	1	1	0
T300	h-	1.416E-09	1.416E-09	5.527E-03	2.870E-08	5.941E-04	1.416E-09	1.416E-09	3.669E-09	3.580E-08	1.166E-02
	p-	1	1	1	1	1	1	1	1	1	1
T400	h-	1.414E-09	1.414E-09	5.474E-02	4.634E-09	1.194E-03	1.414E-09	1.799E-09	4.971E-08	6.143E-07	1.936E-01
	p-	1	1	0	1	1	1	1	1	1	0
T500	h-	1.416E-09	1.416E-09	2.646E-04	1.416E-09	1.043E-02	1.416E-09	1.416E-09	6.184E-08	1.011E-06	4.377E-01
	p-	1	1	1	1	1	1	1	1	1	0
T600	h-	2.286E-09	1.416E-09	9.070E-04	1.597E-09	1.041E-03	1.416E-09	1.416E-09	1.795E-07	5.025E-07	7.562E-01
	p-	1	1	1	1	1	1	1	1	1	0
T700	h-	2.286E-09	1.416E-09	6.795E-07	1.416E-09	2.722E-07	1.416E-09	1.416E-09	8.547E-08	8.095E-06	3.224E-01
	p-	1	1	1	1	1	1	1	1	1	0
T800	h-	2.029E-09	1.416E-09	1.652E-05	1.597E-09	1.230E-06	1.416E-09	1.416E-09	5.559E-07	2.297E-08	9.381E-01
	p-	1	1	1	1	1	1	1	1	1	0
T900	h-	4.638E-09	1.416E-09	3.580E-08	1.800E-09	6.184E-08	1.416E-09	1.416E-09	1.011E-06	2.198E-06	8.159E-01
	p-	1	1	1	1	1	1	1	1	1	0
T1000	h-	2.898E-09	1.416E-09	5.212E-09	1.416E-09	1.167E-08	1.416E-09	1.416E-09	2.054E-08	1.617E-07	2.859E-01
	p-	1	1	1	1	1	1	1	1	1	0
T1200	h-	2.454E-07	1.416E-09	8.281E-09	1.416E-09	4.541E-07	1.416E-09	1.416E-09	6.795E-07	1.814E-06	5.605E-01
	p-	1	1	1	1	1	1	1	1	1	0
T1500	h-	2.777E-05	1.416E-09	6.574E-09	1.416E-09	1.465E-08	1.416E-09	1.597E-09	1.115E-06	9.620E-05	7.415E-01
	p-	1	1	1	1	1	1	1	1	1	0

span value in comparison with the other algorithms. The average of the make-span in each column within Table 8 is calculated and shown in its last row that said that our proposed algorithm, MHMPA, is better with a value of 26.05, MHIMPA comes in the second rank with a value of 26.71, while SMPSO occupies the last rank with an amount of 35.73. MHMPA is considered the best one under all the different VM lengths based on this analysis. It is worth mentioning that the make-span is a phrase about the maximum execution time needed until the last VM finishes execution tasks assigned to it.

Afterward, the average of the carbon dioxide emission rate obtained by each algorithm as the second objective needed to be minimized by our algorithm is calculated and introduced in Table 9. Inspecting this table shows that our proposal is considered the most friendly one to the environment compared to the others. Moreover, the last row in table 9 was introduced to show the average carbon emission rate shown in the other rows for all the different VM lengths (VL). After witnessing the standard obtained by each algorithm shown in this figure, we found that our proposed (MHMPA) is considered the lowest one generated to the carbon emission rate and is the one more friend to the environment.

Furthermore, the flowtime needed by each VM for executing their assigned tasks based on the assignments achieved by each algorithm is shown in Table 10. This table shows that our proposed algorithm (MHMPA) is considered the best one that could assign the tasks to the VM in the form that will reduce the flowtime under all the different VM lengths. In addition, we calculated the average of the flowtime needed based on allocating the tasks using each algorithm to the different VM lengths and introduce in the last row of this table that elaborates the superiority of our proposed algorithm in comparison with the others.

In the end, the average of energy consumed by different VMs lengths under allocation of the tasks using each algorithm is shown in Table 11. According to this table, our proposed could reach less consumed energy in comparison with the others for a number of virtual machines up to 100. However, the proposed couldnŠt outperform DNSGAII for the VMs lengths of 120 and 150. The average of the energy consumed under each algorithm is shown in Table 11. This results shows that our proposal is the one with less energy consumed under any VM length.

Table 12 presents the outcomes of comparing the proposed: MHMPA with the others using the Wilcoxon rank-sum test, it is concluded that the alternative hypothesis is true for most cases and this shows that the outcomes obtained by the proposed algorithm and the others are significantly different.

In Fig. 5, the algorithms are compared in terms of CPU time to show the speedup of each one. This figure shows that our proposed algorithm: MHMPA couldn't overcome OMOPSO, SMPSO, MMPA, MIMPA, and NSGAII as our main limitation tackled in future work.

VI. SUMMARIZATION OF EXPERIMENTS

Finally, in this section, the experiments presented in the previous section will be summarized to show briefly to the readers the performance of each proposed algorithm compared to the competing ones. Generally,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

TABLE 8: Average makespan obtained by MHMPA, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII and MIBEA.

VL	MHMPA	MHIMPA	MIMPA	MMPA	NSGAII	SMPSO	OMOPSO	DMOPSO	NSGAIII	DNSGAII	MIBEA
10	**89.4570**	89.6470	90.5912	91.7943	92.5324	113.1758	101.2761	96.2201	106.943	91.9235	114.0468
20	**45.9691**	46.8855	48.6476	49.1991	47.1491	61.9472	60.62837	52.7048	56.10983	47.2546	61.6710
30	**32.1477**	32.7170	34.2946	36.3018	34.0717	45.2120	41.95644	38.6534	40.82022	34.1940	42.0768
40	**25.1462**	25.9842	27.2273	29.4917	27.4138	35.7520	33.57304	30.4262	31.42105	27.1118	31.9164
50	**20.9689**	22.1296	23.5117	24.9284	23.2689	29.9739	27.9228	26.3831	25.54763	22.9008	26.4609
60	**18.4807**	19.1518	20.5618	21.8721	20.0396	26.5106	24.8255	22.7401	22.09348	20.7188	22.1630
70	**16.2424**	17.1293	18.6113	19.6809	18.3014	23.7714	22.42415	20.5645	19.73083	17.4550	19.6135
80	**14.9177**	15.7937	17.0116	18.4004	16.2405	21.5765	20.10868	18.9452	17.09658	16.4048	17.4486
90	**13.9749**	14.3813	15.6372	16.7380	14.9775	20.1817	18.86657	17.3732	15.64152	14.9188	15.8404
100	**13.0768**	13.5360	14.8082	15.8637	14.0724	18.7703	16.89766	16.1589	14.72178	13.6037	14.5445
120	**11.8814**	12.3502	13.4117	14.1134	12.2931	16.9279	15.37936	14.4464	13.06298	11.7045	12.9750
150	**10.2976**	10.8611	11.6673	5.4020	10.8245	14.9706	13.48678	12.3897	11.33237	10.6723	11.0223
Avg	**26.0467**	26.7139	27.9985	29.1880	27.5987	35.7308	33.1121	30.5838	31.2101	27.4052	32.4816

Bold value indicates the best value

TABLE 9: Average carbon dioxide emission rate obtained from MHMPA, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII, and MIBEA.

VL	MHMPA	MHIMPA	MIMPA	MMPA	NSGAII	SMPSO	OMOPSO	DMOPSO	NSGAIII	DNSGAII	MIBEA
10	**7.560E+05**	7.576E+05	7.626E+05	7.703E+05	7.680E+05	8.508E+05	8.027E+05	7.853E+05	8.250E+05	7.656E+05	8.531E+05
20	**7.675E+05**	7.819E+05	8.037E+05	8.084E+05	7.755E+05	8.960E+05	8.830E+05	8.310E+05	8.465E+05	7.766E+05	8.905E+05
30	**7.882E+05**	8.073E+05	8.350E+05	8.663E+05	8.078E+05	9.511E+05	9.040E+05	8.805E+05	8.884E+05	8.092E+05	9.030E+05
40	**8.092E+05**	8.423E+05	8.690E+05	9.099E+05	8.379E+05	9.874E+05	9.422E+05	9.121E+05	9.020E+05	8.330E+05	9.100E+05
50	8.297E+05	8.745E+05	9.079E+05	9.417E+05	8.661E+05	**1.021E+06**	9.699E+05	9.585E+05	9.116E+05	8.581E+05	9.296E+05
60	8.576E+05	8.961E+05	9.369E+05	9.795E+05	8.839E+05	**1.067E+06**	1.015E+06	9.874E+05	9.312E+05	8.992E+05	9.335E+05
70	8.729E+05	9.207E+05	9.678E+05	**1.009E+06**	9.167E+05	1.104E+06	1.051E+06	1.025E+06	9.569E+05	8.927E+05	9.524E+05
80	8.974E+05	9.543E+05	9.946E+05	**1.050E+06**	9.263E+05	1.137E+06	1.073E+06	1.061E+06	9.537E+05	9.310E+05	9.644E+05
90	9.264E+05	9.700E+05	1.019E+06	**1.070E+06**	9.471E+05	1.180E+06	1.119E+06	1.094E+06	9.698E+05	9.488E+05	9.769E+05
100	9.511E+05	9.974E+05	1.057E+06	**1.113E+06**	9.746E+05	1.214E+06	1.117E+06	1.121E+06	1.008E+06	9.557E+05	9.926E+05
120	**1.007E+06**	1.060E+06	1.119E+06	1.156E+06	1.008E+06	1.293E+06	1.195E+06	1.176E+06	1.044E+06	9.788E+05	1.036E+06
150	**1.068E+06**	1.136E+06	1.195E+06	1.251E+06	1.076E+06	1.400E+06	1.283E+06	1.248E+06	1.106E+06	1.069E+06	1.087E+06
Avg	**8.78E+05**	9.17E+05	9.56E+05	9.94E+05	8.99E+05	1.09E+06	1.03E+06	1.01E+06	9.45E+05	8.93E+05	9.52E+05

Bold value indicates the best value

TABLE 10: Average Flow Time obtained from MHMPA, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII and MIBEA.

VL	MHMPA	MHIMPA	MIMPA	MMPA	NSGAII	SMPSO	OMOPSO	DMOPSO	NSGAIII	DNSGAII	MIBEA
10	23040	**23034**	23073	23105	23327	24514	23673	23347	24300	23205	24760
20	**11744**	11838	12013	11979	11882	12829	12766	12165	12498	11890	12950
30	**8036**	8128	8279	8436	8236	8907	8781	8476	8751	8241	8863
40	**6188**	6342	6434	6572	6487	6889	6886	6578	6772	6406	6815
50	**5092**	5240	5334	5438	5290	5729	5651	5489	5580	5298	5668
60	**4401**	4472	4608	4694	4623	4947	4887	4664	4841	4664	4823
70	**3823**	3954	4072	4124	4057	4372	4364	4151	4277	4021	4281
80	**3477**	3583	3653	3715	3644	3894	3890	3725	3776	3711	3840
90	**3186**	3262	3348	3404	3370	3609	3583	3401	3487	3346	3508
100	**2950**	2997	3125	3175	3123	3326	3262	3132	3236	3076	3237
120	**2614**	2655	2753	2771	2814	2908	2897	2745	2884	2678	2905
150	**2247**	2309	2374	2401	2350	2515	2501	2337	2519	2391	2467
Avg	**6400**	6485	6589	6651	6600	7037	6928	6684	6910	6577	7010

Bold value indicates the best value

Fig. 6 and 7 present the average of CDER, FT, MK, and consumed energy fulfilled by each algorithm under various task lengths and a fixed-VM length of 50. Inspecting those two figures affirms occupying the first rank by MHMPA for all performance metrics: CDER, energy, MK, and FT under various ask lengths, while DNSGAII occupied the last rank for CDER, energy, and FT, and NSGAIII for MK. Generally, MHMPA proved its efficiency for finding the taskŠs assignment which optimizes the various employed performance metrics. In addition, to see the ability of the proposed algorithms and the competing ones in assigning the tasks in a way that will minimize the different performance metrics when the VM lengths are increased, under various VM lengths ranging between 10 and 150, the average of those metrics based on the outcomes of each algorithm has been calculated and displayed in Fig. 8 and 9 which also affirms the efficiency of our proposed algorithm: MHMPA for all metrics. Notwithstanding the superiority of the proposed MHMPA, it suffers from consuming a computational cost a little higher than some of the competing algorithms as our main

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

TABLE 11: Average energy obtained from MHMPA, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII and MIBEA

VL	MHMPA	MHIMPA	MIMPA	MMPA	NSGAII	SMPSO	OMOPSO	DMOPSO	NSGAIII	DNSGAII	MIBEA
10	**3.212E+05**	3.219E+05	3.240E+05	3.273E+05	3.272E+05	3.615E+05	3.410E+05	3.337E+05	3.505E+05	3.253E+05	3.625E+05
20	**3.261E+05**	3.322E+05	3.415E+05	3.435E+05	3.295E+05	3.807E+05	3.752E+05	3.531E+05	3.597E+05	3.300E+05	3.784E+05
30	**3.349E+05**	3.430E+05	3.548E+05	3.681E+05	3.440E+05	4.041E+05	3.841E+05	3.741E+05	3.775E+05	3.438E+05	3.837E+05
40	**3.438E+05**	3.579E+05	3.692E+05	3.866E+05	3.586E+05	4.196E+05	4.003E+05	3.876E+05	3.833E+05	3.539E+05	3.866E+05
50	**3.525E+05**	3.716E+05	3.858E+05	4.001E+05	3.635E+05	4.338E+05	4.121E+05	4.073E+05	3.873E+05	3.646E+05	3.950E+05
60	**3.644E+05**	3.808E+05	3.981E+05	4.162E+05	3.765E+05	4.533E+05	4.314E+05	4.195E+05	3.956E+05	3.820E+05	3.967E+05
70	**3.709E+05**	3.912E+05	4.112E+05	4.286E+05	3.826E+05	4.690E+05	4.467E+05	4.353E+05	4.066E+05	3.793E+05	4.047E+05
80	**3.813E+05**	4.055E+05	4.226E+05	4.460E+05	3.912E+05	4.832E+05	4.560E+05	4.510E+05	4.052E+05	3.956E+05	4.098E+05
90	**3.936E+05**	4.122E+05	4.331E+05	4.545E+05	3.990E+05	5.014E+05	4.754E+05	4.650E+05	4.121E+05	4.031E+05	4.151E+05
100	**4.041E+05**	4.238E+05	4.491E+05	4.727E+05	4.154E+05	5.158E+05	4.745E+05	4.765E+05	4.249E+05	4.061E+05	4.217E+05
120	4.277E+05	4.503E+05	4.754E+05	4.911E+05	4.396E+05	5.495E+05	5.077E+05	4.997E+05	4.437E+05	**4.159E+05**	4.401E+05
150	4.538E+05	4.825E+05	5.075E+05	5.316E+05	4.472E+05	5.950E+05	5.450E+05	5.305E+05	4.701E+05	**4.535E+05**	4.617E+05
Avg	**3.729E+05**	3.894E+05	4.060E+05	4.222E+05	3.812E+05	4.639E+05	4.375E+05	4.278E+05	4.014E+05	3.794E+05	4.047E+05

Bold value indicates the best value

TABLE 12: Wilcoxon Rank sum test between MHMPA, MHIMPA, MIMPA, MMPA, NSGAII, SMPSO, OMOPSO, DMOPSO, NSGAIII, DNSGAII, and MIBEA based on the makespan under the various VM lengths.

VL		DMOPSO	SMPSO	NSGAII	NSGAIII	DNSGAII	OMOPSO	MIBEA	MMPA	MIMPA	MHIMPA
10	p-	1.416E-09	1.416E-09	1.597E-09	1.416E-09	9.158E-07	1.416E-09	1.416E-09	2.574E-09	1.647E-06	1.116E-01
	h-	1	1	1	1	1	1	1	1	1	0
20	h-	1.416E-09	1.416E-09	7.856E-03	1.416E-09	4.785E-04	1.416E-09	1.416E-09	4.638E-09	7.380E-09	9.620E-05
	p-	1	1	1	1	1	1	1	1	1	1
30	h-	1.416E-09	1.416E-09	9.696E-06	1.416E-09	1.268E-05	1.416E-09	1.416E-09	1.416E-09	4.638E-09	2.145E-05
	p-	1	1	1	1	1	1	1	1	1	1
40	h-	1.416E-09	1.416E-09	2.054E-08	1.416E-09	1.647E-06	1.416E-09	1.416E-09	1.597E-09	2.574E-09	2.454E-07
	p-	1	1	1	1	1	1	1	1	1	1
50	h-	1.416E-09	1.416E-09	6.891E-08	1.416E-09	3.996E-08	1.416E-09	1.416E-09	1.416E-09	2.297E-08	1.230E-06
	p-	1	1	1	1	1	1	1	1	1	1
60	h-	1.416E-09	1.416E-09	3.017E-07	2.029E-09	4.638E-09	1.416E-09	1.416E-09	1.416E-09	6.574E-09	1.268E-05
	p-	1	1	1	1	1	1	1	1	1	1
70	h-	1.416E-09	1.416E-09	3.313E-04	1.416E-09	1.671E-03	1.416E-09	1.416E-09	1.416E-09	2.898E-09	1.194E-03
	p-	1	1	1	1	1	1	1	1	1	1
80	h-	1.416E-09	1.416E-09	5.910E-05	9.288E-09	2.273E-04	1.416E-09	3.261E-09	1.416E-09	7.380E-09	7.393E-06
	p-	1	1	1	1	1	1	1	1	1	1
90	h-	1.416E-09	1.416E-09	5.559E-07	3.261E-09	7.139E-07	1.416E-09	2.286E-09	1.800E-09	5.855E-09	1.652E-05
	p-	1	1	1	1	1	1	1	1	1	1
100	h-	2.574E-09	1.416E-09	1.302E-01	6.795E-07	3.320E-01	1.416E-09	5.618E-06	4.638E-09	2.454E-07	1.624E-01
	p-	1	1	0	0	0	1	1	1	1	0
120	h-	1.800E-09	1.416E-09	5.866E-03	1.495E-06	2.072E-01	1.416E-09	3.017E-07	2.054E-08	1.465E-08	6.850E-04
	p-	1	1	1	1	0	1	1	1	1	1
150	h-	1.416E-09	1.416E-09	5.941E-04	7.509E-07	1.373E-02	1.416E-09	1.617E-07	1.416E-09	2.286E-09	6.159E-06
	p-	1	1	1	1	1	1	1	1	1	1

limitation addressed in the future.

VII. CONCLUSION AND FUTURE WORK

Fog computing (FC) plays a crucial role in improving the quality of services (QoS) to several applications, especially healthcare. QoS include improving the response time until the real-time applications could quickly take their decision and at the same time as the events. The tasks offloaded using the IoT must be distributed among the fog nodes accurately until utilizing them optimally to reduce the response time, energy consumption, carbon dioxide emission rate, and flow time. According to the need for optimizing more than one objective at the same time rather than one objective at a time like most of the recently published paper did, in this paper, marine predators algorithm-based multi-objective approach and integrated with the polynomial mutation operator to increase its diversification capability is proposed for tackling the MTSFC under minimizing both the make-span and the carbon dioxide emission ratio. This proposed algorithm was abbreviated as MHMPA. Additionally, the effect of using the Cauchy distribution with the levy flight instead of the Gaussian distribution on the performance of the standard marine predators algorithm was investigated; this improved version is abbreviated as MIMPA. The main advantage of the Cauchy distribution is that it could increase the exploitation capability by giving a small ratio to the exploration capability to avoid stuck into local minima. MIMPA and MHMPA use the Pareto optimality to find all the solutions that trade-off between the make-span and the carbon dioxide emission rate, in addition to using an external archive to save those solutions. Experimentally, MIMPA and MHMPA are validated and compared with many well-known robust multiobjective algorithms: NSGAII, NSGAIII,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

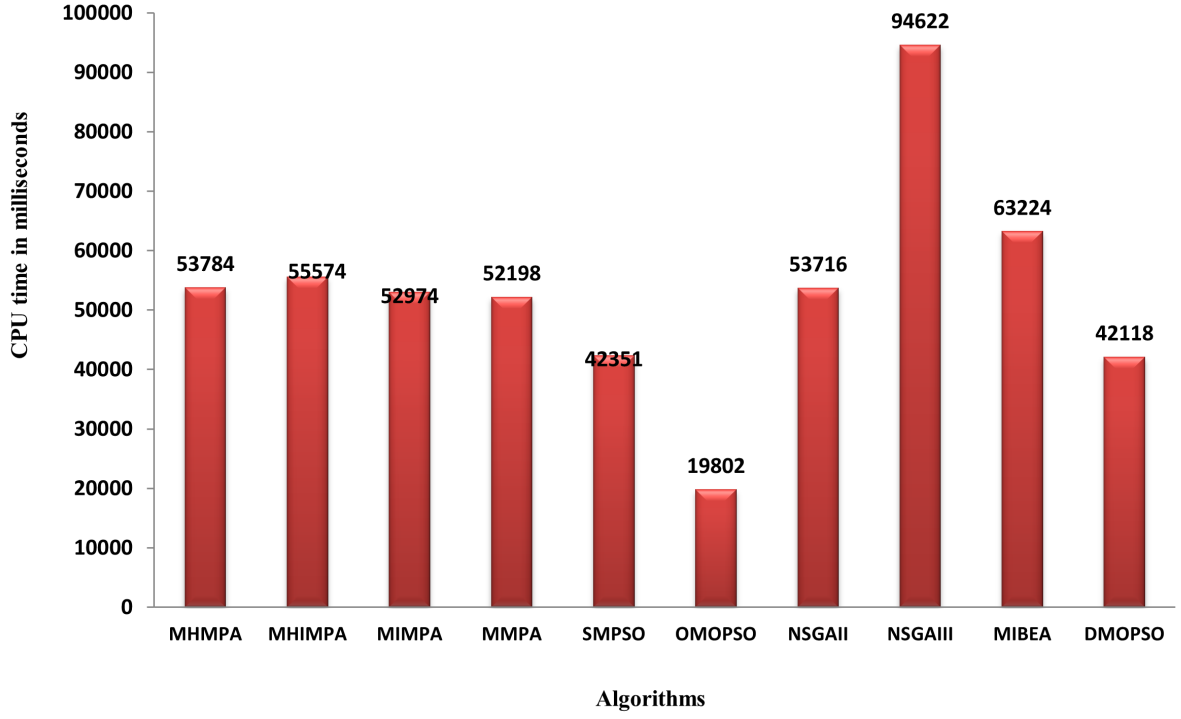


FIGURE 5: Comparison among the algorithms based on CPU time.

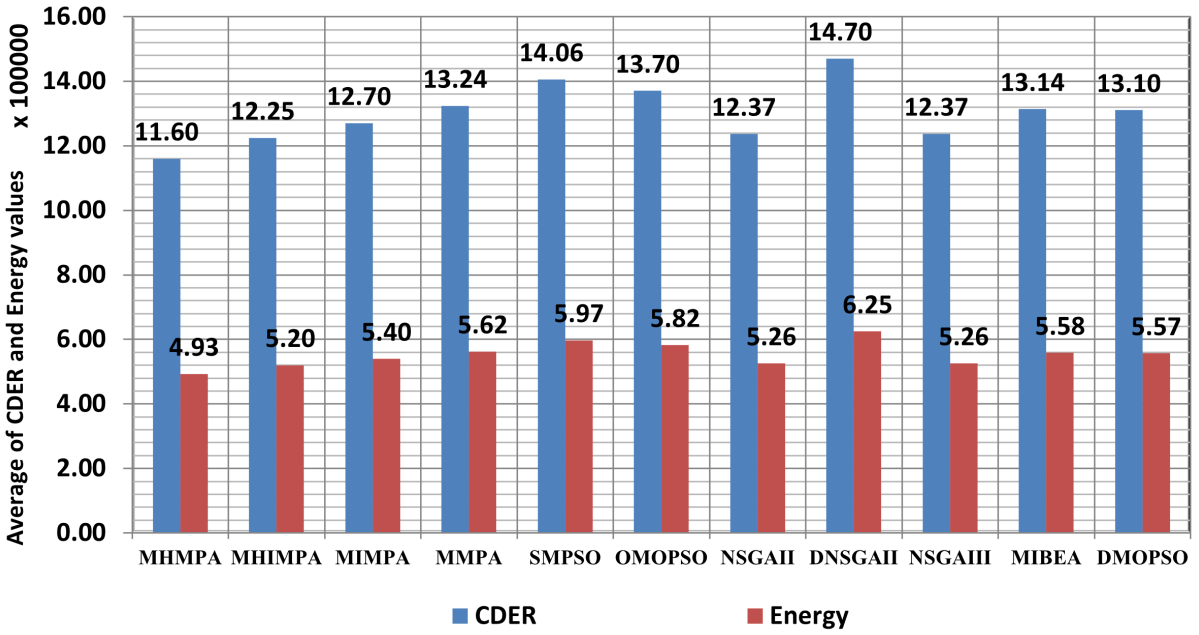


FIGURE 6: Comparison among the algorithms based on CDER and Energy under various task lengths and fixed-VM length.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3111130, IEEE Access

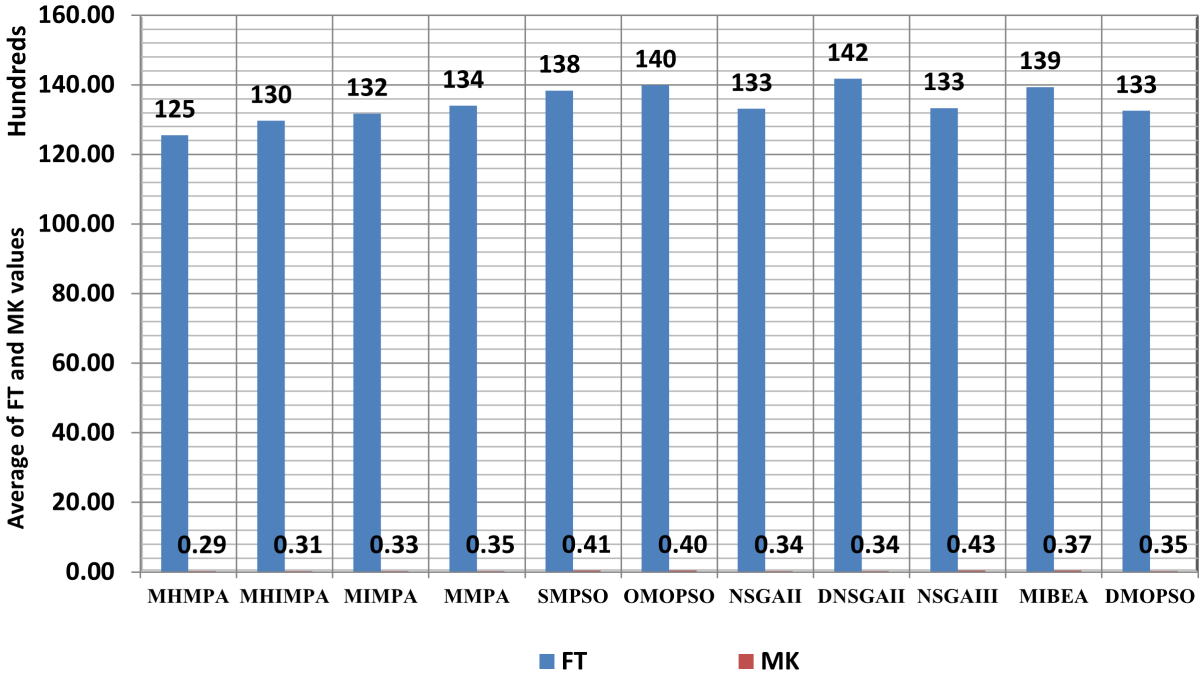


FIGURE 7: Comparison among the algorithms based on FT and MK under various task lengths and fixed-VM length.

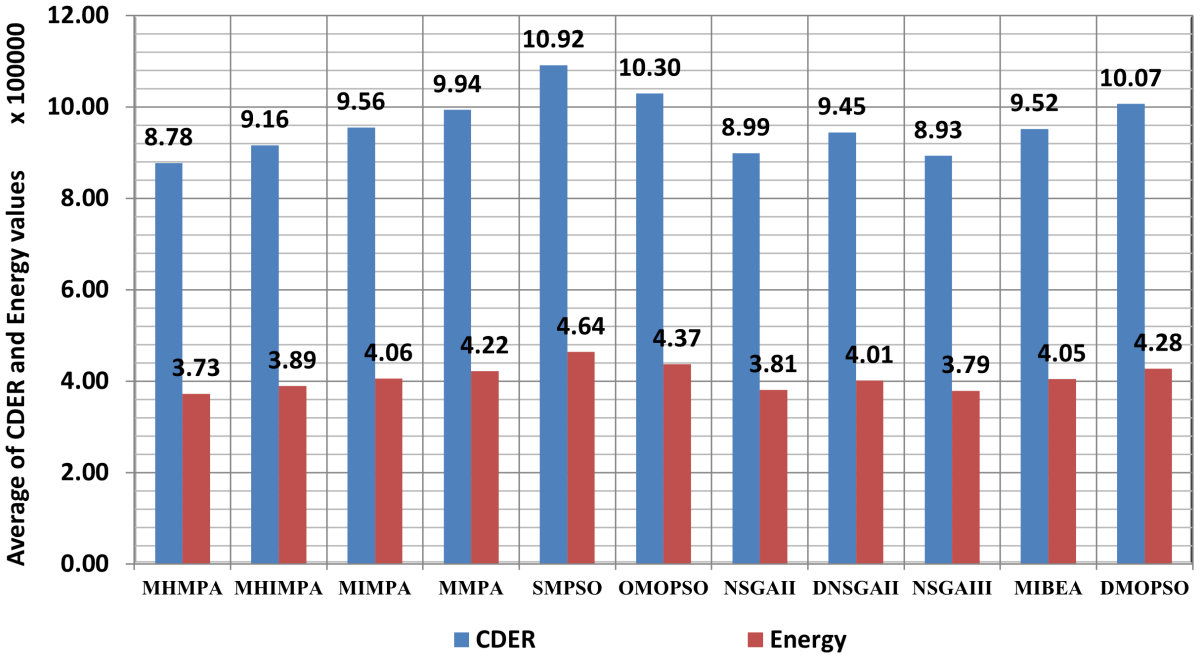


FIGURE 8: Comparison among the algorithms based on CDER and Energy under various VM lengths and fixed-task length.

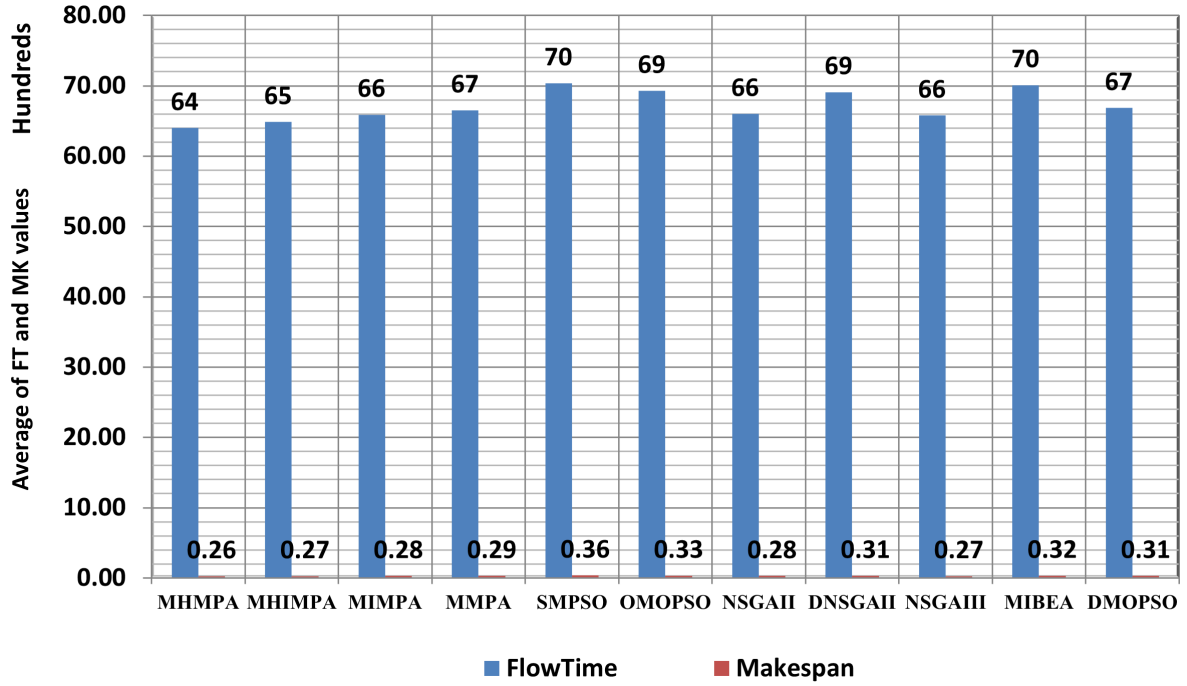


FIGURE 9: Comparison among the algorithms based on FT and MK under various VM lengths and fixed-task length

.

SMPSO, OMOPSO, DMOPSO, DNSGAII, MIBEA, and MMPA under four metrics: make-span, flow time, carbon emission rate, and energy. The experiment outcomes show that MIMPA could be better under those metrics in comparison to the standard one and most other compared algorithms. But, MHMPA could outperform all compared algorithms, involving MIMPA, under various metrics but CPU time, which is our main limitation tackled in future work. Our future work also includes applying the IMPA for overcoming several other problems: DNA fragment assembly problem, 0-1 knapsack problem, solar photovoltaic parameter estimation problem, and Image segmentation problem.

REFERENCES

[1] A. A. Mutlag, M. K. Abd Ghani, N. a. Arunkumar, M. A. Mohammed, and O. Mohd, "Enabling technologies for fog computing in healthcare iot systems," Future Generation Computer Systems, vol. 90, pp. 62–78, 2019.

[2] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Towards fault tolerant fog computing for iot-based smart city applications," in 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0752–0757, IEEE, 2019.

[3] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," Journal of Systems Architecture, vol. 98, pp. 289–330, 2019.

[4] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in 2012 Proceedings IEEE Infocom, pp. 945–953, IEEE, 2012.

[5] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 49–62, 2010.

[6] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing, pp. 15–20, 2013.

[7] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," IEEE Communications Magazine, vol. 54, no. 9, pp. 22–28, 2016.

[8] X. Wei, J. Liu, Y. Wang, C. Tang, and Y. Hu, "Wireless edge caching based on content similarity in dynamic environments," Journal of Systems Architecture, vol. 115, p. 102000, 2021.

[9] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud ran," IEEE Transactions on Wireless Communications, vol. 15, no. 9, pp. 6118–6131, 2016.

[10] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," IEEE Journal on Selected Areas in Communications, vol. 34, no. 8, pp. 2275–2284, 2016.

[11] M. Gregori, J. Gómez-Vilardebó, J. Matamoros, and D. Gündüz, "Wireless content caching for small cell and d2d networks," IEEE Journal on Selected Areas in Communications, vol. 34, no. 5, pp. 1222–1234, 2016.

[12] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," IEEE Journal on Selected Areas in Communications, vol. 34, no. 1, pp. 176–189, 2015.

[13] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Energy-efficient design for edge-caching wireless networks: When is coded-caching beneficial?," in 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 1–5, IEEE, 2017.

[14] T. X. Vu, S. Chatzinotas, B. Ottersten, and T. Q. Duong, "Energy minimization for cache-assisted content delivery networks with wireless backhaul," IEEE Wireless Communications Letters, vol. 7, no. 3, pp. 332–335, 2017.

[15] C. Yang, Y. Yao, Z. Chen, and B. Xia, "Analysis on cache-enabled

wireless heterogeneous networks," IEEE Transactions on Wireless Communications, vol. 15, no. 1, pp. 131–145, 2015.

[16] K. Yu, Z. Ma, R. Ni, and T. Zhang, "A caching strategy based on many-to-many matching game in d2d networks," Tsinghua Science and Technology, vol. 26, no. 6, pp. 857–868, 2021.

[17] N. Garg and T. Ratnarajah, "Cooperative scenarios for multi-agent reinforcement learning in wireless edge caching," in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3435–3439, IEEE, 2021.

[18] Y. Fu, Z. Shi, J. Ke, H. Wang, A. K. Wong, and T. Q. Quek, "Efficient delay minimization algorithm for cache-enabled noma systems," IEEE Wireless Communications Letters, 2021.

[19] Y. Fu, Q. Yu, A. K. Wong, Z. Shi, H. Wang, and T. Q. Quek, "Exploiting coding and recommendation to improve cache efficiency of reliability-aware wireless edge caching networks," IEEE Transactions on Wireless Communications, 2021.

[20] T. Zhang, S. Biswas, and T. Ratnarajah, "On the performance of cache-enabled hybrid wireless networks," IEEE Transactions on Communications, vol. 69, no. 3, pp. 1818–1834, 2020.

[21] N. Garg and T. Ratnarajah, "Tensor completion based prediction in wireless edge caching," in 2020 54th Asilomar Conference on Signals, Systems, and Computers, pp. 1579–1582, IEEE, 2020.

[22] Q. Wang and S. Chen, "Latency-minimum offloading decision and resource allocation for fog-enabled internet of things networks," Transactions on Emerging Telecommunications Technologies, p. e3880, 2020.

[23] M. Abdel-Basset, D. El-Shahat, K. Deb, and M. Abouhawwash, "Energy-aware whale optimization algorithm for real-time task scheduling in multiprocessor systems," Applied Soft Computing, p. 106349, 2020.

[24] M. Abdel-Basset, V. Chang, and R. Mohamed, "A novel equilibrium optimization algorithm for multi-thresholding image segmentation problems," Neural Computing and Applications, pp. 1–34, 2020.

[25] M. Abdel-Basset, R. Mohamed, M. Elhoseny, R. K. Chakrabortty, and M. Ryan, "A hybrid covid-19 detection model using an improved marine predators algorithm and a ranking-based diversity reduction strategy," IEEE Access, vol. 8, pp. 79521–79540, 2020.

[26] M. Allaoui, B. Ahiod, and M. El Yafrani, "A hybrid crow search algorithm for solving the dna fragment assembly problem," Expert Systems with Applications, vol. 102, pp. 44–56, 2018.

[27] M. Abdel-Basset, R. Mohamed, M. Abouhawwash, R. K. Chakrabortty, and M. J. Ryan, "Ea-msca: An effective energy-aware multi-objective modified sine-cosine algorithm for real-time task scheduling in multiprocessor systems: Methods and analysis," Expert systems with applications, vol. 173, p. 114699, 2021.

[28] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," Enterprise Information Systems, vol. 12, no. 4, pp. 373–397, 2018.

[29] C. Guerrero, I. Lera, and C. Juiz, "Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures," Future Generation Computer Systems, vol. 97, pp. 131–144, 2019.

[30] D. Rahbari and M. Nickray, "Scheduling of fog networks with optimized knapsack by symbiotic organisms search," in 2017 21st Conference of Open Innovations Association (FRUCT), pp. 278–283, IEEE, 2017.

[31] N. Javaid, A. A. Butt, K. Latif, and A. Rehman, "Cloud and fog based integrated environment for load balancing using cuckoo levy distribution and flower pollination for smart homes," in 2019 International Conference on Computer and Information Sciences (ICCIS), pp. 1–6, IEEE, 2019.

[32] X.-S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," Engineering computations, 2012.

[33] M. K. Hussein and M. H. Mousa, "Efficient task offloading for iot-based applications in fog computing using ant colony optimization," IEEE Access, vol. 8, pp. 37191–37201, 2020.

[34] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, and C. Maple, "A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing," IEEE Access, vol. 7, pp. 115760–115773, 2019.

[35] S. Wang, T. Zhao, and S. Pang, "Task scheduling algorithm based on improved firework algorithm in fog computing," IEEE Access, vol. 8, pp. 32385–32394, 2020.

[36] M. Ghobaei-Arani, A. Souri, F. Safara, and M. Norouzi, "An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing," Transactions on Emerging Telecommunications Technologies, vol. 31, no. 2, p. e3770, 2020.

[37] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-aware marine predators algorithm for task scheduling in iot-based fog computing applications," IEEE Transactions on Industrial Informatics, 2020.

[38] S. H. H. Madni, M. S. Abd Latiff, J. Ali, et al., "Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds," Arabian Journal for Science and Engineering, vol. 44, no. 4, pp. 3585–3602, 2019.

[39] A. Abebe, Developing a Hybrid Heuristic and Metaheuristic Scheduling Algorithm (HHMH) for Cloud-Fog Computing Environment. PhD thesis, ASTU, 2021.

[40] S. S. Mohammadi and M. Deypir, "A new meta-heuristic algorithm based on tabu search for the job scheduling problem in a fog-cloud system," Journal of Modeling in Engineering, vol. 18, no. 62, 2020.

[41] B. Thisarasinghe and K. Jayasena, "Optimized task scheduling on fog computing environment using meta heuristic algorithms," 2019.

[42] N. Téllez, M. Jimeno, A. Salazar, and E. Nino-Ruiz, "A tabu search method for load balancing in fog computing," Int. J. Artif. Intell, vol. 16, no. 2, 2018.

[43] A. A. Butt, S. Khan, T. Ashfaq, S. Javaid, N. A. Sattar, and N. Javaid, "A cloud and fog based architecture for energy management of smart city by using meta-heuristic techniques," in 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 1588–1593, IEEE, 2019.

[44] M. A. Soliman, H. M. Hasanien, and A. Alkuhayli, "Marine predators algorithm for parameters identification of triple-diode photovoltaic models," IEEE Access, vol. 8, pp. 155832–155842, 2020.

[45] M. Ebeed, A. Alhejji, S. Kamel, and F. Jurado, "Solving the optimal reactive power dispatch using marine predators algorithm considering the uncertainties in load and wind-solar generation systems," Energies, vol. 13, no. 17, p. 4316, 2020.

[46] P.-H. Dinh, "A novel approach based on three-scale image decomposition and marine predators algorithm for multi-modal medical image fusion," Biomedical Signal Processing and Control, vol. 67, p. 102536, 2021.

[47] N. Wang, J. Wang, L. Zhu, H. Wang, and G. Wang, "A novel dynamic clustering method by integrating marine predators algorithm and particle swarm optimization algorithm," IEEE Access, vol. 9, pp. 3557–3569, 2020.

[48] X. Sun, G. Wang, L. Xu, H. Yuan, and N. Yousefi, "Optimal performance of a combined heat-power system with a proton exchange membrane fuel cell using a developed marine predators algorithm," Journal of Cleaner Production, vol. 284, p. 124776, 2021.

[49] A. A. Z. Diab, M. A. Tolba, A. G. A. El-Magd, M. M. Zaky, and A. M. El-Rifaie, "Fuel cell parameters estimation via marine predators and political optimizers," IEEE Access, vol. 8, pp. 166998–167018, 2020.

[50] Q. Fan, H. Huang, Q. Chen, L. Yao, K. Yang, and D. Huang, "A modified self-adaptive marine predators algorithm: framework and engineering applications," Engineering with Computers, pp. 1–26, 2021.

[51] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine predators algorithm: A nature-inspired metaheuristic," Expert Systems with Applications, p. 113377, 2020.

[52] M. Abdel-Basset, D. El-shahat, M. Elhoseny, and H. Song, "Energy-aware metaheuristic algorithm for industrial internet of things task scheduling problems in fog computing applications," IEEE Internet of Things Journal, 2020.

[53] Y.-J. Zhang, Z. Liu, H. Zhang, and T.-D. Tan, "The impact of economic growth, industrial structure and urbanization on carbon emission intensity in china," Natural hazards, vol. 73, no. 2, pp. 579–595, 2014.

[54] H.-J. Yang and J. He, "Analysis of the ethnic minority living activities co2 emissions from 2012 to 2014: A case study in yunnan," in Advanced Materials and Energy Sustainability: Proceedings of the 2016 International Conference on Advanced Materials and Energy Sustainability (AMES2016), pp. 611–620, World Scientific, 2017.

[55] D. Sarkar and J. M. Modak, "Pareto-optimal solutions for multi-objective optimization of fed-batch bioreactors using nondomi-

nated sorting genetic algorithm," Chemical Engineering Science, vol. 60, no. 2, pp. 481–492, 2005.

[56] K. Deb and M. Abouhawwash, "An optimality theory-based proximity measure for set-based multiobjective optimization," IEEE Transactions on Evolutionary Computation, vol. 20, no. 4, pp. 515–528, 2015.

[57] Y. Liu and B. Cao, "A novel ant colony optimization algorithm with levy flight," IEEE Access, vol. 8, pp. 67205–67213, 2020.

[58] K. Deb and S. Tiwari, "Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization," European Journal of Operational Research, vol. 185, no. 3, pp. 1062–1087, 2008.

[59] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," IEEE transactions on evolutionary computation, vol. 6, no. 2, pp. 182–197, 2002.

[60] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints," IEEE transactions on evolutionary computation, vol. 18, no. 4, pp. 577–601, 2013.

[61] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. C. Coello, F. Luna, and E. Alba, "Smpso: A new pso-based metaheuristic for multi-objective optimization," in 2009 IEEE Symposium on computational intelligence in multi-criteria decision-making (MCDM), pp. 66–73, IEEE, 2009.

[62] M. R. Sierra and C. A. C. Coello, "Improving pso-based multi-objective optimization using crowding, mutation andâĹĹ-dominance," in International conference on evolutionary multi-criterion optimization, pp. 505–519, Springer, 2005.

[63] S. Zapotecas Martínez and C. A. Coello Coello, "A multi-objective particle swarm optimizer based on decomposition," in Proceedings of the 13th annual conference on Genetic and evolutionary computation, pp. 69–76, 2011.

[64] X. Cai, H. Sun, and Z. Fan, "A diversity indicator based on reference vectors for many-objective optimization," Information Sciences, vol. 430, pp. 467–486, 2018.

[65] W. Li, E. Özcan, R. John, J. H. Drake, A. Neumann, and M. Wagner, "A modified indicator-based evolutionary algorithm (mibea)," in 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1047–1054, IEEE, 2017.

[66] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," Advances in Engineering Software, vol. 42, no. 10, pp. 760–771, 2011.

[67] W. Haynes, "Wilcoxon rank sum test," Encyclopedia of systems biology, pp. 2354–2355, 2013.