Energy-Aware Metaheuristic Algorithm for Industrial-Internet-of-Things Task Scheduling Problems in Fog Computing Applications

Mohamed Abdel-Basset, *Member, IEEE*, Doaa El-Shahat, Mohamed Elhoseny, *Senior Member, IEEE*, and Houbing Song, *Senior Member, IEEE*

***Abstract*—In Industrial-Internet-of-Things (IIoT) applications, fog computing (FC) has soared as a means to improve the Quality of Services (QoSs) provided to users through cloud computing, which has become overwhelmed by the massive flow of data. Transmitting all these amounts of data to the cloud and coming back with a response can cause high latency and requires high network bandwidth. The availability of sustainable energy sources for FC servers is one of the difficulties that the service providers can face in IIoT applications. The most important factor contributing to energy consumption on fog servers is task scheduling. In this article, we suggest an energy-aware metaheuristic algorithm based on a Harris Hawks optimization algorithm based on a local search strategy (HHOLS) for task scheduling in FC (TSFC) to improve the QoSs provided to the users in IIoT applications. First, we describe the high virtualized layered FC model taking into account its heterogeneous architecture. The normalization and scaling phase aids the standard Harris hawks algorithm to solve the TSFC, which is discrete. Moreover, the swap mutation ameliorates the quality of the solutions due to its ability to balance the workloads among all virtual machines. For further improvements, a local search strategy is integrated with HHOLS. We compare HHOLS with other metaheuristics using various performance metrics, such as energy consumption, makespan, cost, flow time, and emission rate of carbon dioxide. The proposed algorithm gives superior results in comparison with other algorithms.**

***Index Terms*—Carbon dioxide emission rate (CDER), energy, fog computing (FC), makespan, metaheuristic, task scheduling.**

I. Introduction

FOG COMPUTING (FC) has emerged as a means to support the Quality of Services (QoSs) provided to users through cloud computing. Fog nodes are not rich in computing resources so they are used along with cloud computing. Until recently, cloud computing has been the perfect choice for many users to manage, process, and archive their big data in remote data centers (DCs) through the Internet. Unfortunately, the demand for cloud computing services has been increased to exploit their enormous computing resources with a comfortable pricing model that follows pay-as-you-go. The excessive pressure on cloud computing services leads to high latency and requires a high-bandwidth network. The high latency caused by the cloud can turn grace into a curse. For example, processing a reliable real-time data is crucial for medical wearables as it tracks the health of patients. Consequently, the service providers are forced to expand the DCs or create new ones to fulfill the ambitions of their users and ensure the QoS offered.

The rise in energy consumption, electricity, costs, and the emission rate of carbon dioxide is the tax of expanding the DCs. Also, the more running servers in these DCs, the more cooling and lightning equipment are required. That is why we had to provide an appropriate environment for DCs, including choosing the proper location as well as climatic conditions to maintain the performance of these servers. The most fearful thing is that the volume of stored data in DCs, client devices, and machine to machine (M2M) will be triple increased by 2021 from 1.8 in 2016 to 7.2 ZB [1]. Also, in 2021, it is not only the people who produce the data, but the machines and things have a large amount of data, estimated at 850 ZB. Only 85 ZB of this huge data is useful while neglecting the remaining data. Approximately 10% of 85 ZB will be stored or used (7.2 ZB in 2021). The useful data can exceed the DC traffic by a factor of four.

We need to look for ways to mitigate energy consumption to process that vast amount of data, such as digitalization, green DCs, and renewable energy. Digitalization is a new trend to use digital technologies to retain the sustainability of energy systems, safety, and productivity [2]. The investment in digital electricity reached $47 billion in 2016. Green DCs have appeared to make use of energy-efficient technologies, including low power servers, free air cooling, and smart grid. The service providers will be forced to search for new ways to power their DCs with renewable energy, such as solar, wind, and geothermal power.

Lately, FC has come to extend the cloud computing capabilities to low latency, location awareness, support mobility, and real-time interaction. IoT is one of the most emerging concepts that have been well integrated with FC to include healthcare [3], smart city [4], smart manufacturing [5], and traffic [6]. IoT collects the data from several sensors and

Manuscript received April 10, 2020; revised June 16, 2020 and July 3, 2020; accepted July 24, 2020. Date of publication July 30, 2020; date of current version August 6, 2021. *(Corresponding author: Mohamed Elhoseny.)*

Mohamed Abdel-Basset and Doaa El-Shahat are with the Department of Computer Science, Zagazig University, Zagazig 44519, Egypt (e-mail: analyst_mohamed@zu.edu.eg; doaazedan@zu.edu.eg).

Mohamed Elhoseny is with the Faculty of Computers and Information, Mansoura University, Dakahlia 35516, Egypt (e-mail: mohamed_elhoseny@mans.edu.eg).

Houbing Song is with the Department of Electrical Engineering and Computer Science, Embry–Riddle Aeronautical University, Daytona Beach, FL 32114 USA (e-mail: h.song@ieee.org).

Digital Object Identifier 10.1109/JIOT.2020.3012617

smart devices and sends this data for processing in FC, which reduces the latency that previously done through transmitting massive data with high bandwidth to cloud in remote centralized DCs. Later, if the data needed to be stored, it will be sent to the cloud DCs for long-term storage and analytics.

Most of the energy consumed in the FC environment comes from the servers, network, storage, and infrastructure (cooling and lighting equipment) [2]. According to the International Energy Agency (IEA), the global energy demands of DCs reached approximately 194 Terawatt-hours (TWh) in 2014 [7]. By 2020, the IEA expects that the energy demands will reach only about 200 TWh thanks to the continued efficiency gains. The energy consumed by the servers will be expected to increase from 86.4 to 107.7 TWh, which will occupy about 54% of the total energy consumed. The servers are powered by 39% of petroleum products, 24% of natural gas, 23% of coal, 8% of nuclear, and 6% of other products [8]. The different mix of products can generate different carbon dioxide emission rates (CDERs). The CO_2 emission in gram (g) for each TWh of various sources, is presented in [9]. It is worth mentioning that the coal without scrubbing has the highest CO_2 emission rate with a value of 1050 g/Kwh.

Energy consumption can be a barrier to the performance of FC. Scheduling tasks on fog servers is a complex and challenging problem, as inefficient task scheduling can result in higher energy consumption. It can cause performance degradation of the energy system. It increases air pollution and greenhouse gas emissions. Sooner or later, the increase in energy consumption has serious consequences. It threatens the existence of nonrenewable energy sources, such as coal, natural gas, gasoline, and oil, which in turn led to higher consumer energy bills.

The motivation behind developing an energy-aware technique for task scheduling in FC (TSFC) comes from the aforementioned consequences of TSFC and the success of the Harris hawk optimization algorithm in solving various real-world problems. The main contributions of this article are as follows.

1) The adopted virtualized FC system is introduced and a new metaheuristic algorithm Harris Hawks optimization algorithm based on a local search strategy (HHOLS) is proposed for tackling TS of IoT applications in FC.
2) Normalization and scaling phase is essential to make the algorithm to deal with the discrete nature of TSFC.
3) The proposed algorithm adopted a swap mutation operation and a local search strategy to balance the workload of tasks among all virtual machines (VMs) and improve the quality of the best solution.
4) Our proposed algorithm HHOLS is compared with other algorithms based on makespan, energy consumption, cost, flow time, CDER, and fitness function.

The remainder of this article is structured as follows. Section II presents the related work done for the TSFC problem. Section III introduces a description of the fog system model while illustrating the TSFC problem. Furthermore, Section IV presents the standard Harris hawks optimization (HHO) algorithm. In Section V, the proposed algorithm (HHOLS) is further investigated and elaborated. In Section VI, we provide an illustrative example of TSFC. Section VII gives numerical and comparison results. Finally, we introduce some conclusions and future work in Section VIII.

II. Related Work

Recently, TSFC has attracted the attention of many researchers. Wu and Lee [10] proposed an energy minimization scheduling (EMS) algorithm for reducing the energy consumption for IoT workflows on distributed FC. Li *et al.* [11] provided a comprehensive virtualization model in FC to keep pace with the evolution of IoT. Khattak *et al.* [12] distributed the load between the servers in the FC to efficiently use the computing resources. Also, a limited amount of data will be sent to the cloud, reducing latency because tolerance is unacceptable in healthcare systems.

Pham and Huh [13] proposed a heuristic-based algorithm for task scheduling in the cloud-FC to achieve the balance between the makespan and the costs of cloud resources. Yin *et al.* [14] built a new task scheduling model based on the characteristics of containers for smart manufacturing. Tran *et al.* [15] have developed an approach that can enhance the performance of IoT services in terms of response time, cost, and energy. Xu *et al.* [16] presented a virtual machine scheduling method to avoid overload and low-load of resource usage. Rahbari and Nickray [17] handled the task scheduling problem in which requests are locally processed in FC for multiple regions. The heuristic methods may find difficulties to solve TS in complex fog systems [18].

Metaheuristic algorithms play a pivotal role in solving TSFC [19]–[21]. Bitam *et al.* [22] proposed a bees life algorithm for scheduling the tasks in FC concerning the CPU execution time and the allocated memory. Guerrero *et al.* [23] have compared the performance of three evolutionary algorithms to optimize the network latency and resources used for the service placement in FC. Rahbari and Nickray [24] presented a knapsack-based scheduling in fog networks using the symbiotic organisms' search algorithm. Javaid *et al.* [25] designed the cuckoo search and flower pollination algorithms for load balancing of the incoming requests in the cloud, which helps to overcome the delay and latency caused by cloud and improves the fog performance. These studies above focus on balancing the workloads of tasks and CPU execution time.

Jayasena and Thisarasinghe [26] used the whale optimization algorithm for TSFC in comparison with round robin, shortest job first, and PSO algorithms. Moreover, in [27], the moth-flame optimization algorithm is applied for TSFC. The algorithm needs to be more explored with the load balancing strategies and takes the makespan and the transfer and execution times into consideration. Wang *et al.* [28] proposed an improved firework algorithm that improves the processing time and load balancing among the tasks. In addition, Hosseinioun *et al.* [29] incorporated an invasive weed optimization with the culture evolutionary algorithm for TSFC. Lately, Mishra *et al.* [30] have developed the BAT, PSO, and binary PSO algorithms for service allocation in

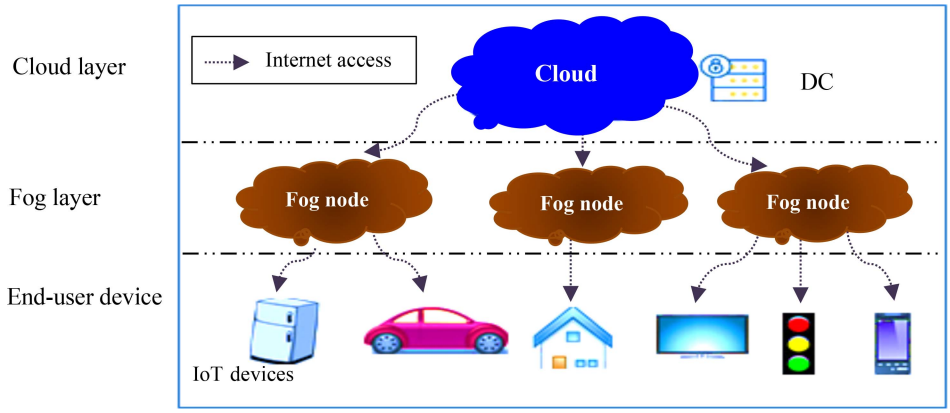

Fig. 1. Architecture of the fog system model.

FC with different performance metrics: makespan, energy consumption, flow time, cost. First, the tasks are assigned randomly to the VMs, and the solutions are ameliorated through iterations. However, the quality of the solutions is improved through iterations. We find that this improvement is not sufficient, and there is still a load imbalance.

However, numerous metaheuristics are developed for TSFC, these metaheuristics may face one or more of these limitations, such as not considering load balancing strategies for the tasks, the need to be compared with more metaheuristics, and not regarding some performance aspects, including makespan, flow time, energy consumption, costs, and CDER. Our proposed algorithm HHOLS comes to overcome loads of unbalancing problem using swap mutation and local search strategy. Also, HHOLS is compared with the most recent and powerful metaheuristics regarding the different performance metrics (makespan, flow time, energy consumption, costs, and CDER).

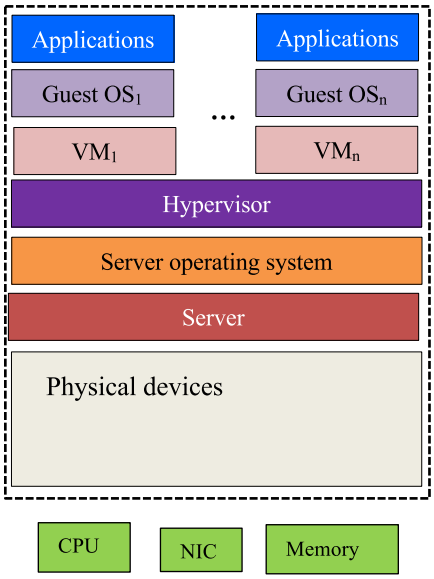

Fig. 2. Server virtualization.

III. Task Scheduling in Fog System

FC is an intermediate layer between the cloud layer and the end-user device layer, as in Fig. 1. It is a computing platform with a large number of distributed nodes, which is highly virtualized and scalable. It brings the cloud computing services to the edge of the network to achieve low latency, low cost, and incorporation of IoT devices. Each fog node offers cloud computing resources (servers, storage, network, and applications) as a service through Internet access.

The fog server (physical machine) is virtualized to take advantage of reduced costs, faster provisioning of resources, and easier disaster recovery. The hypervisor or virtual machine monitor programs allow creating and managing new VMs. There are n VMs that can run on a single fog server (VM_1, $VM_2, \ldots, VM_n$). All VMs share the physical resources of fog server, such as memory, disks, network interface controller (NIC), and CPU cores. Moreover, these resources can be effectively scaled up or down with minimal interaction from the service providers, according to the service level agreement (SLA). Each virtual machine has its own guest operating system (OS), which can run various heterogeneous applications, as depicted in Fig. 2.

Task scheduling is one of the thorniest problems we face in FC. The problem arises when there is a set of heterogeneous m tasks needed to be scheduled on n heterogeneous VMs. The tasks are assumed to be independent of each other as they are submitted by several independent users to the fog.

The task scheduling on the FC can be summarized as follows. There is a set of m tasks t_i, where $i = 1, 2, \ldots, m$. Each task t_i has an associated workload w_i. The workload of a task can be expressed in terms of Million Instructions (MI). The tasks are said to be heterogeneous if they have different workloads. There is a set of n VMs VM_j, where $j = 1, 2, \ldots, n$. Each virtual machine VM_j has CPU processing elements, memory, network bandwidth, and storage. Each virtual machine VM_j has a processing speed s_j in terms of MI Per Seconds (MIPS). Each task t_i can be assigned to only one virtual machine VM_j at a time. The expected time to compute (ETC) contains the expected execution time e_{ij} of each task t_i on each virtual machine VM_j as in Table I.

The execution time e_{ij} of each task includes the time of computation added to the communication time. The variation

TABLE I
ETC MATRIX

$m \times n$	VM_1	VM_2	...	VM_n
t_1	e_{11}	e_{12}	...	e_{1n}
t_2	e_{21}	e_{22}	...	e_{2n}
...	...	...	...	e_{ij}
...	...	...	...	e_{ij}
t_m	e_{m1}	e_{m2}	...	e_{mn}

of execution times for a task in a row of ETC matrix on different machines represents the machine heterogeneity. Similarly, the variation of execution times for a set of tasks in a column of the ETC matrix on a single virtual machine represents task heterogeneity. The ETC matrix can be calculated by

$$e_{ij} = \frac{w_i}{s_j}. \tag{1}$$

IV. HARRIS HAWKS OPTIMIZATION ALGORITHM

Haidari *et al.* [31] proposed a nature-inspired algorithm that simulates the behaviors of Harris hawks called HHO algorithm as they have a surprising collective action in the way they track and pounce on their prey. HHO algorithm performed the exploration and exploitation stages through investigating the prey, sudden pounce, and using various attacking techniques. In the exploration stage, Harris' hawks are randomly distributed to locations in anticipation of prey using two strategies. Harris' hawks are considered the candidate solutions, and the best solution is the one which is the purposed prey or near the optimum. Harris' hawks perch based on two strategies regarding the location of other family members and the rabbit and on random tall trees with an equal chance of q for each as follows:

$$y(t+1) = \begin{cases} y_r(t) - r_1|y_r(t) - 2r_2 y(t)|, & q \geq 0.5 \\ y_{\text{rabbit}}(t) - y_{\text{mean}}(t) - r_3(L + r_4(U - L)), & q < 0.5 \end{cases} \tag{2}$$

where $y(t)$ and $y(t+1)$ indicate the position vectors of hawks in the current and next iteration, respectively. $y_r(t)$ is a random hawk selected from the population. $y_{\text{rabbit}}(t)$ is the position of the rabbit. q, r_1, r_2, r_3, and r_4 are randomly generated numbers. L and U are lower and upper bounds to generate random locations inside hawks' home. $y_{\text{mean}}(t)$ is the mean position of hawks in the current population which can be calculated as

$$y_{\text{mean}}(t) = \frac{1}{h}\sum_{i=1}^{h} y_i(t) \tag{3}$$

where $y_i(t)$ is the position vector of each hawk indexed by i in the population at iteration t and $i = 1, \ldots, h$. Based on the escaping energy E of the rabbit, HHO can turn from the exploration to the exploitation phase as follows:

$$E = 2E_0\left(1 - \frac{t}{\text{Max\_iter}}\right) \tag{4}$$

where E_0 is the rabbit initial energy which is randomly generated in $[-1, 1]$. *Max_iter* determines the maximum number of iterations. When $|E| \geq 1$, hawks look for more regions to explore the rabbit location, otherwise, the exploitation stage will occur. With an equal chance p, the success ($p \geq 0.5$) or failure ($p < 0.5$) of rabbit escape is formulated in the HHO algorithm. According to the rabbit energy, the hawks will perform a soft besiege if ($|E| \geq 0.5$) or a hard one if ($|E| < 0.5$) one. In the soft besiege, the hawks encircle the rabbit softly and then they suddenly pounce the rabbit. It can be mathematically modeled as

$$y(t+1) = \triangle y(t) - E|J * y_{\text{rabbit}}(t) - y(t) \tag{5}$$

where $\triangle y(t)$ is the difference between the hawk and rabbit positions. J refers to the random jump strength of the rabbit and through the escaping process and is drawn using a random number rand $\in [0, 1]$. $\triangle y(t)$ and J can be calculated as follows:

$$\triangle y(t) = y_{\text{rabbit}}(t) - y(t) \tag{6}$$

$$J = 2(1 - \text{rand}). \tag{7}$$

In the other hand, the hard besiege can be formulated as

$$y(t+1) = y_{\text{rabbit}}(t) - E|\triangle y(t)|. \tag{8}$$

Soft besiege with progressive rapid dives (PRDs) is happened when ($|E| \geq 0.5$) and ($p < 0.5$) as the rabbit has the chance of successful escape. The hawks have the ability to choose the best possible dive. Lévy flight is used to simulate the leapfrog of the prey. The next move of the hawks is evaluated to decide if the previous dive is good using

$$k = y_{\text{rabbit}}(t) - E|Jy_{\text{rabbit}}(t) - y(t)|. \tag{9}$$

If the previous dive is not good, the hawks will dive using Lévy flight L pattern as follows:

$$z = k + s \times L(d) \tag{10}$$

where d is the problem dimension and s is a random vector with size d. *Levy* can be calculated by

$$\text{Levy} = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \tag{11}$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\left(\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}\right)}\right)^{\frac{1}{\beta}} \tag{12}$$

u and v are random numbers $\in [0, 1]$.β is a constant set to 1.5. The final soft besiege is updated using

$$y(t+1) = \begin{cases} k, & \text{if } f(k) < f(y(t)) \\ z, & \text{if } f(z) < f(y(t)) \end{cases} \tag{13}$$

where k and z are calculated using (9) and (10). Hard besiege with PRD is happened when ($|E| < 0.5$) and ($p < 0.5$) as the rabbit has not enough energy to escape using (13) where z is calculated using (10) and k is updated using the following equations:

$$k = y_{\text{rabbit}}(t) - E|Jy_{\text{rabbit}}(t) - y_{\text{mean}}(t)|. \tag{14}$$

The pseudocode of the HHO algorithm is presented in Algorithm 1.

Algorithm 1: Standard HHO Algorithm

Input: population size h and maximum number of iterations *Max_iter*
Initialize a population of h random hawks $y_i(i = 1, \ldots, h)$
Calculate the fitness of each hawk
y_{rabbit} = best position of minimum fitness
$t = 1$
while $(t \leq Max\_iter)$
  Update E using Eq. (4)
  if $(|E| \geq 1)$
  Update $y(t+1)$ using Eq. (2)
  end if
  If $(|E| < 1)$
    if $(p \geq 0.5 \;\&\&|E| \geq 0.5)$
      Update $y(t+1)$ using soft besiege
    end if
    if $(p \geq 0.5 \;\&\&|E| < 0.5)$
      Update $y(t+1)$ using hard besiege
    end if
    if $(p < 0.5\&\&|E| \geq 0.5)$
      Update $y(t+1)$ using soft besiege with PRD
    end if
    if $(p < 0.5\&\&|E| \geq 0.5)$
      Update $y(t+1)$ using hard besiege with PRD
    end if
  end if
  if(fitness($y(t+1)$)< fitness (y_{rabbit}))
    Update $y_{rabbit} = y(t+1)$
  end if
  $t++$
end while
Output: rabbit position y_{rabbit}

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
3	1	2	1	3	3	2	2

Fig. 3. Harris hawk vector representation.

V. Proposed Algorithm

We proposed a HHO algorithm based on a local search strategy (HHOLS) that ameliorates the performance of the standard HHO algorithm for solving the task scheduling problem of IIOT applications in FC. The proposed algorithm is composed of five main steps, including initialization, evaluation, normalization and scaling, swap mutation, and local search strategy. In the following sections, we will clearly explain each step of the algorithm.

A. Initialization

In this phase, a random population of h hawks is generated. HHO algorithm has been designed to deal with continuous problems, but TSFC is formulated as a discrete problem. Each hawk is represented by a vector of size m which is the number of tasks submitted by the users to the fog node to be scheduled. Then, a random number $\in [1, \ n]$ is randomly generated where n is the number of VMs so that the VMs are randomly assigned to the tasks. If we have a task scheduling problem with $m = 8$ and $n = 3$, then a candidate solution can be considered as shown in Fig. 3. We find that VM_3 is assigned to t_1, VM_1 is assigned to t_2, and so on.

B. Evaluation

Five performance measures (makespan, energy, cost, flow time, and CDER) used for evaluating the solutions in the initial population. Also, we will explain a fitness function used to assess the solutions. We will review each of them separately.

1) Makespan: Tasks are independent of each other. Once a VM has finished processing a task, another task begins, until it finishes processing all assigned tasks. All the VMs are assumed to start to process their tasks at time 0. Accordingly, the execution time Et_j for each virtual machine VM_j is the completion time of the last processed task assigned to the machine VM_j. In addition to the solution vector, an allocation matrix is considered to allocate the tasks to VMs. Each task t_i is allocated or not allocated to a specific virtual machine VM_j according to a decision variable A_{ij} as follows:

$$A_{ij} = \begin{cases} 1, & \text{if } t_i \text{ is allocated to } VM_j \\ 0, & \text{if } t_i \text{ isn't allocated to } VM_j. \end{cases} \tag{15}$$

Finally, the makespan MK is the maximum total execution time of a virtual machine VM_j among all VMs

$$MK = \max Et_j, \quad j = 1, \ldots, n \tag{16}$$

$$Et_j = \sum_{i=1}^{m} A_{ij} \times e_{ij} \tag{17}$$

where Et_j is the total execution time of a virtual machine VM_j.

2) Energy: Each virtual machine can be in an active or idle state. The energy consumption of the server is about 60% of its active state. Thus, the energy consumed by VM_j is the energy consumed in the active state as well as in the idle one. The idle time of VM_j is the execution time of VM_j subtracted from the makespan. The energy of each VM can be calculated as

$$\text{Energy}\big(VM_j\big) = \big(Et_j \times \text{beta}_j + \big(MK - Et_j\big) * \text{alpha}_j\big) \times s_j \tag{18}$$

$$\text{beta}_j = 10^{-8} \times \left(s_j^2\right) \tag{19}$$

$$\text{alpha}_j = 0.6 * \text{beta}_j \tag{20}$$

where beta_j is the energy consumed in terms of joules per MI in the active state of each VM. The total energy consumed in the FC system is the summation of energy consumed by all VMs which can be computed as

$$\text{total\_energy} = \sum_{\text{j}=1}^{\text{n}} \text{Energy}\big(\text{VM}_\text{j}\big). \tag{21}$$

3) Cost: The cost of renting the server resources (CPU, NIC, memory, and disks) is an important measure of performance to reduce the costs. Because VMs have a heterogeneous structure, different VMS are given different costs depending on their capabilities. To compute the cost of processing all tasks assigned to the different VMs, the cost can be computed as

$$\text{total\_cost} = \sum_{j=1}^{n} \text{cost}_j \times ET_j. \tag{22}$$

The total cost is the summation of recruitment costs for all VMs in the FC system. While VM cost is the time of execution

ET_j to complete all assigned tasks multiplied by the price cost_j of server resource rentals per unit of time.

4) Flow Time: Flow time is another important performance measure that gives a measure of time that a task spends within the fog system. It is the sum of finishing times of all tasks in their VMs that can be calculated as

$$\text{flow\_time} = \sum_{i=1}^{n} f_i. \tag{23}$$

5) Carbon Dioxide Emission Rate: As mentioned previously, the different mix of energy sources that can be used to power the servers can generate different CDERs. The CDER caused by energy consumption can be computed as

$$\text{CDER} = \sum_{k=1}^{4} \text{total\_energy} \times sh_k \times Em\_\text{factor}_k \times \text{ratio}. \tag{24}$$

The power is a mix of coal, oil, natural gas, and nonfossil products that are numbered as $k = 1, 2, 3, 4$, respectively. sh_k represents the share of the kth energy source in the total energy consumption. Moreover, Em_factor_k is the emission factor of the kth energy source. It can be concluded that the total energy consumption is directly proportional to the carbon emission rate. If the amount of the consumed energy decreases, the CO_2 emission rate will decrease. 0.7476, 0.5825, 0.4435, and 0 are the carbon emission factors for coal, oil, natural gas, and nonfossil products, respectively [32]. *ratio* refers to the conversion ratio from carbon to carbon dioxide and is set to 44/12 [33].

6) Fitness Function: Makespan and energy consumption are two of the most important factors influencing the cost, carbon dioxide rate, and time flow. Thus, a biobjective function is employed for evaluating the candidate solutions as follows:

$$\text{fitness} = \tau \times \text{total\_energy} + (1 - \tau) \times MK \tag{25}$$

where $\tau = 0.8$, as we pay more attention to energy consumption.

C. Normalization and Scaling Phase

In this phase, the newly generated hawk $y(t + 1)$ of HHO contains continuous values. The continuous values of hawk must be converted to discrete values (VMs numbers). First, the hawk vector is normalized in the interval [0, 1] as follows:

$$\text{normalized}_i(t + 1) = \frac{y_i(t + 1) - \min}{\max - \min} \tag{26}$$

where *min* and *max* are the minimum value and the maximum value in the hawk vector, respectively. $y_i(t+1)$ represents each hawk value where $i = 1, \ldots, m$. $\text{normalized}_i(t + 1)$ refers to the ith value in the normalized hawk vector. After that, using the following equation, the normalized hawk will be scaled in [1, n] as represented in Fig. 4:

$$\text{scaled}_i(t + 1) = \text{normalized}_i(t + 1) * (n - 1) + 1 \tag{27}$$

where $\text{scaled}_i(t+1)$ is the ith value in the scaled hawk vector. n is the number of VMs.

Algorithm 2: Local Search Strategy

Input: the best solution y_{rabbit}
$y_{improved\_rabbit} = y_{rabbit}$
Compute total execution time ET_jOf each VM_j using Eq. (17)
Find makespan using Eq. (16)
VM_{MK}= stores VM index accompanied with makespan
Find minimum ET_j
$VM_{\text{min}ET}$ = stores VM index accompanied by a minimum ET_j
Select a random task t_{rand} assigned to VM_{MK} where *rand* $\in[1,m]$
Remove t_{rand} assigned to VM_{MK}
Assign t_{rand} to $VM_{\text{min}ET}$
if (fitness ($y_{improved\_rabbit}$) < fitness(y_{rabbit}))
$y_{rabbit} = y_{improved\_rabbit}$
end if
Output: y_{rabbit}

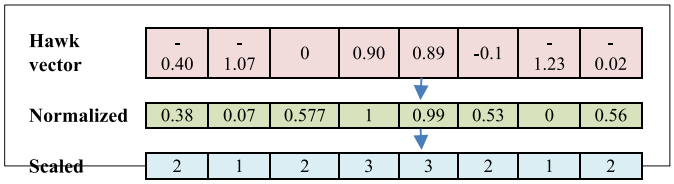

Fig. 4. Application of normalization and scaling phase for a continuous hawk vector.

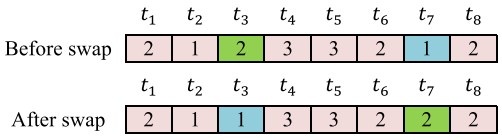

Fig. 5. Swap mutation operation.

D. Swap Mutation

Swap mutation is used to improve the quality of candidate solutions in the population per iteration. Two random locations are selected in the candidate solution. The two locations are switched only if the swap reduces the fitness value; otherwise, there will be no exchange. In Fig. 5, the locations 3 and 7 are randomly chosen to be swapped as the result of a new candidate solution has a minimum fitness value. After mutation, VM_2 will run t_7 and VM_1 will run t_3.

E. Local Search Strategy

For further improvements, a local search strategy is utilized to prop the quality of the best solution per iteration. The pseudocode is presented in Algorithm 2. To relieve the workload of tasks assigned to VM_jaccompanying the makespan (maximum ET_j). A random task assigned to VM_j accompanying the makespan will be removed. It will be assigned to VM_j with minimum ET_j. The task will be removed from a VM to another if it lessens the fitness value. This strategy helps to balance the workload among VMs that minimizes the makespan. Finally, the pseudocode of the proposed algorithm HHOLS is introduced in Algorithm 3.

Algorithm 3: The Proposed Algorithm HHOLS

Input: population size h, Max_iter, n, m, s_j of VMs, w_i of tasks
Initialize a population of h random hawks $y_i(i = 1, \ldots, h)$
Calculate the fitness of each hawk using Eq. (25)
Find the best position with minimum fitness y_{rabbit}
$t = 1$
while $(t \leq Max\_iter)$
  Update E using Eq. (4)
  if $(|E| \geq 1)$
  Update $y(t+1)$ using Eq. (2)
  end if
  If $(|E| < 1)$
    if $(p \geq 0.5 \&\& |E| \geq 0.5)$
      Update $y(t+1)$ using soft besiege
    end if
    if $(p \geq 0.5 \&\& |E| < 0.5)$
      Update $y(t+1)$ using hard besiege
    end if
    if $(p < 0.5 \&\& |E| \geq 0.5)$
      Update $y(t+1)$ using soft besiege with PRD
    end if
    if $(p < 0.5 \&\& |E| \geq 0.5)$
      Update $y(t+1)$ using hard besiege with PRD
    end if
  end if
  Apply Normalization and scaling phase toy$(t+1)$
  Apply swap operation to y$(t+1)$
  if (fitness $y(t+1)$ < fitness(y_{rabbit}))
    Update $y_{rabbit} = y(t+1)$
  end if
  Apply local search strategy to y_{rabbit}
  $t++$
end while
Output: rabbit position y_{rabbit}

TABLE II
DESCRIPTION OF TASKS

t_i	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
w_i	3000	5600	4800	7600	9800	10000	2000	8100

TABLE III
DESCRIPTION OF VMS

VM_j	VM_1	VM_2	VM_3
VM speed	1000	2000	2500
Cost	100	120	200

VI. ILLUSTRATIVE EXAMPLE

A TSFC problem is described in Tables II and III with $m = 8$ and $n = 3$. Each task has a workload in terms of MI. Each virtual machine has a processing speed in terms of MIPS and a cost. We will consider a candidate solution (3-1-2-1-3-3-2-2) in our explanation. We follow the next steps to solve a TSFC problem.

Step 1: We compute the ETC matrix using (1) as in Table IV.

Step 2: We calculate the corresponding allocation matrix as in Table V using (15).

Step 3: We compute the execution time ET_j of each VM using (17) as presented in Table VI.

Step 4: We use (16) to compute the makespan. The Gantt chart in Fig. 6 depicts the finish time of each task on the corresponding virtual machine. We can see that the makespan is 13.2, which comes from VM_1.

Step 5: We compute beta_j, alpha_j, $\text{Energy}(VM_j)$ using (18), (19), and (21), respectively. The details of energy consumption for all VMs are introduced in Table VII. Then, the total energy is 2811.5 using (21).

Step 6: The cost is 4038 using (22).

Step 7: The flow time is 47.489 using (23).

Step 8: To compute the CDER, we must first know the composition of energy sources and the share of each product. It was assumed that the energy sources are coal, oil, natural gas, and nonfossil products. Table VIII presents the emission factor and the share [33] of each product. The carbon emission rate is 6617.0174.

TABLE IV
CALCULATION OF ETC MATRIX

8×3	VM_1	VM_2	VM_3
t_1	3	1.5	1.2
t_2	5.6	2.8	2.24
t_3	4.8	2.4	1.92
t_4	7.6	3.8	3.04
t_5	9.8	4.9	3.92
t_6	10	5	4.0
t_7	2	1	0.8
t_8	8.1	4.05	3.24

TABLE V
ALLOCATION MATRIX

8×3	VM_1	VM_2	VM_3
t_1	0	0	1
t_2	1	0	0
t_3	0	1	0
t_4	1	0	0
t_5	0	0	1
t_6	0	0	1
t_7	0	1	0
t_8	0	1	0

TABLE VI
EXECUTION TIME OF ALL VMS

1×3	VM_1	VM_2	VM_3
ET_j	13.2	7.45	9.12

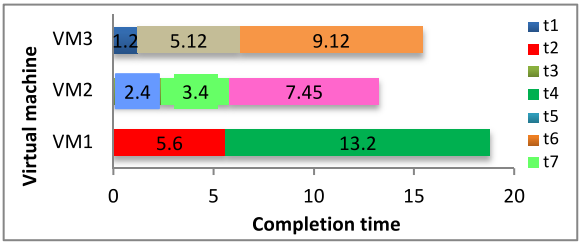

Fig. 6. Gantt chart for the candidate solution (3-1-2-1-3-3-2-2).

TABLE VII
ENERGY CONSUMPTION OF VMS

	VM_1	VM_2	VM_3
$beta_j$	0.01	0.04	0.0625
$alpha_j$	0.006	0.024	0.0375
$Energy(VM_j)$	132	872	1807

TABLE VIII
DESCRIPTION OF ENERGY SOURCES

Product	Coal	Oil	Natural gas	Non-fossil products
Emission factor	0.7476	0.5825	0.4435	0
Percentage	68.4%	18.6%	5%	8%

Step 9: The fitness value of the candidate solution is 2251.84 which can be calculated using (25).

VII. RESULTS AND DISCUSSION

In this section, we will shed light on the performance of the proposed algorithm in comparison with other algorithms. The proposed algorithm HHOLS is encoded in Java and run on a laptop with the following specifications: Intel Core i5-3317U CPU @ 1.70 GHz with 4.0-GB Memory in Windows 10 with a 64-b OS.

A. Data Set Description

The data set is randomly generated based on two principles. In the first principle, the different sizes of heterogeneous tasks are run on a fixed number of VMs. In the second principle, the different sizes of heterogeneous VMs run a fixed number of tasks. The two principles are used for further studying the impact of increasing VMs or tasks on the different performance measures (makespan, energy consumption, costs, flow time, CDER, and fitness).

We generate ten tasks with different lengths (100, 200, 300, 400, 500, 600,700, 800, 900, and 1000) to fulfill the first principle, and the number of VMS is 50. The workloads of each task are heterogeneous and randomly generated within the interval [1000, 10 000]. The 50 VMs were varying in terms of different processing speeds. The processing speed of the first 25 VMs is set to 1000 MIPS, while the processing speed of the remaining 25 is set to 2000 MIPS. The expenses of using the VMs should be proportional to their processing speed, so the cost of the first 25 VMs is set to \$100 per time unit, but \$150 per time unit for the remaining 25 VMs.

We generate ten VMs with different lengths (10, 20, 30, 40, 50, 60, 70, 80, 90, and 100), and the length of tasks is 500 to fulfill the second principle. The processing speed of the VMs is randomly generated in [1000, 10000] to ensure VM heterogeneity. For each VM length, the processing speed of the first half is 1000, while the processing speed of the second half is 2000. The cost of the first half is \$100 per unit time, whereas the cost of the remaining half is \$150 per unit time.

B. Parameter Setting

The performance of the proposed algorithm is verified by doing comparisons with other state-of-the-art algorithms. Our selection of these algorithms has been adopted because they are the most modern and powerful as they have proven successful in many applications [34]–[38]. These algorithms are summarized as follows: flower algorithm [39] based on the local search strategy (FALS), particle swarm optimization [40] based on the local search strategy (PSOLS), multiverse optimizer algorithm [41] based on the local search strategy (MVALS), gray wolf optimization algorithm [42] based on the local search strategy (GWOLS), whale algorithm [43] based on the local search strategy (WALS), and bat algorithm [44] based on the local search strategy (BALS).

TABLE IX
PARAMETER SETTING OF THE ALGORITHMS

Algorithm	Parameter	Value
FALS	λ	1.5
	p	0.8
PSOLS	$C1$	1
	$C2$	1
	$Weight\_inertia$	0.9
MVALS	min	0.2
	max	1
	$Exploitation\_accuracy$	6
GWOLS	a	Decreasing in [2, 0]
WALS	b	1
BALS	Maximum frequency	0
	Maximum frequency	2
	loudness	0.5
	pulse rate	0.4
HHOLS	λ	1.5

For an accurate and fair comparison, we apply the same modifications (swap mutation and local search strategy) for all the algorithms. For all the next experiments, the maximum number of iterations is 1500 as we note that the improvement after 1500 is slow. τ is set to 0.8 as our primary concern is how to lessen energy consumption. The number of individuals within the population is 10. We evaluate each algorithm by 20 independent runs. Setting parameter values of the algorithms can affect the performance of the algorithms. Analyzing the effect of parameter tuning requires a large number of practical experiments. Finally, the parameter settings that can obtain the best results are informed in Table IX.

C. Comparison Among Algorithms Using Different Virtual Machines Lengths

We conduct an experiment that includes the proposed algorithm and the other six algorithms (FALS, BALS, PSOLS, MVALS, GWOLS, and WALS) for performance investigation. Fig. 7 brings a comparison among algorithms based on the average fitness for different VMs lengths and 500 tasks. This experiment shows that HHOLS succeeds in obtaining lower fitness values compared with the other algorithms. Also, we can see that the increase in VM lengths results in maximizing the fitness values. But on the other hand, if we look at the values of the makespan, we will note that it is decreasing by increasing the lengths of VMs, as seen in Fig. 8. We can conclude that the fitness values increase, whereas the makespan values decrease as the VMs length increase. The increase in fitness values come from the increase in energy consumption that is related to increasing VMs lengths. Running more VMs in the fog system consumes much energy. The fitness value is affected by 80% of the energy consumption value plus 20% of the makespan value.

In Fig. 9, a comparison-based total_energy$_{avg}$, which represents the total energy consumption on average used by the fog system to run 500 tasks on all different VMs lengths.

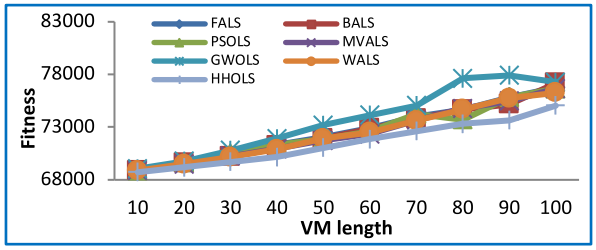

Fig. 7. Comparison based on the average fitness for 500 tasks and different VMs lengths.

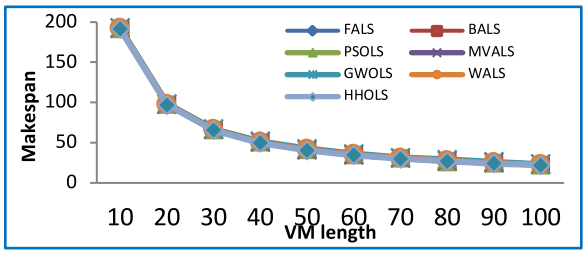

Fig. 8. Comparison based on the average makespan for 500 tasks and different VMs lengths.

total_energy$_{\text{avg}}$ can be computed as

$$\text{total\_energy}_{\text{avg}} = \sum_{j=1}^{M}\left(\sum_{i=1}^{N}\text{total\_energy}_{ji} \times \frac{1}{N}\right) \quad (28)$$

where M is the number of VMs lengths (here, $M = 10$). N is the number of evaluations for each instance of VM length (here, $N = 20$). total_energy$_{ji}$ is the obtained total energy by the jth instance of a specific virtual machine length in the ith evaluation. From the figure, we find that the proposed algorithm HHOLS achieves the lowest energy consumption compared with the remainder of the algorithms. In Fig. 10, the cost of using VMs has been significantly reduced when comparing HHOLS to the other algorithms. The proposed algorithm has the lowest values of cost for all different lengths of VMs. Fig. 11 shows us the total emission rate of CO_2 resulted from running the ten different lengths of VMs and 500 tasks for each algorithm. The total emission rate in average case is computed as

$$\text{CDER}_{\text{avg}} = \sum_{k=1}^{4}\text{total\_energy}_{\text{avg}} \times sh_k \times Em\_\text{factor}_k \times \text{ratio} \quad (29)$$

where total_energy$_{\text{avg}}$ is obtained by (28).

D. Comparison Among Algorithms Using Different Task Lengths

In this experiment, we study the effect of increasing lengths of tasks when the number of VMs is fixed. The number of VMs is 50. The number of task lengths is ranging from 100 to1000. In Fig. 12, a comparison-based average makespan is provided to study the effect of increasing the lengths of tasks whereas the number of VMs is fixed. We can infer that increasing the task lengths maximizes the makespan due to

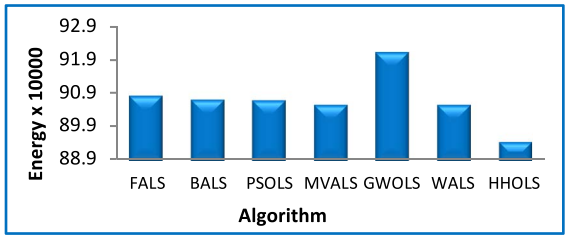

Fig. 9. Comparison based on the total energy in the average case for 500 tasks and different VMs lengths.

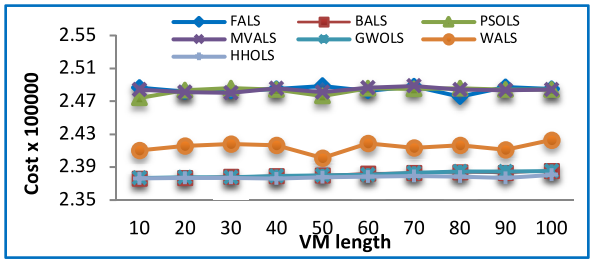

Fig. 10. Comparison based on the average cost for 500 tasks and different VMs lengths.

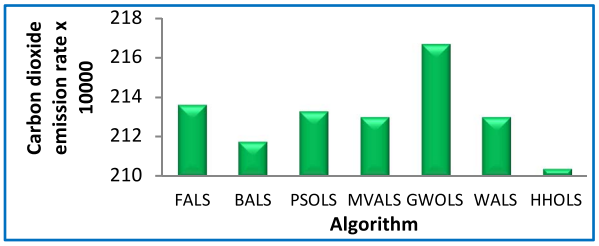

Fig. 11. Total CDERs in the average case for 500 tasks and different VMs lengths.

the increased pressure and workload on the existing VMS, which delays the completion of tasks. Also, we can see that HHOLS has the minimum makespan in most of the different task lengths among all algorithms.

Moreover, Fig. 13 depicts the total energy consumed in the fog system to run all the different lengths of tasks. It shows us that HHOLS is the energy-aware algorithm as it obtains the minimum total energy. Consequently, saving energy consumption affects the carbon dioxide emissions rate and costs. Fig. 14 proves that the proposed algorithm is a friend to the environment because of the CO_2 emission rate reduction compared to the other algorithms. HHOLS comes in the rank with minimum CO_2 emission rate value. From Fig. 15, we can see that HHOLS obtains the highest average CUP time with a value of 8.2 s for solving all the data sets, while MVALS obtains the minimum CPU time with a value of 5.2 s. Although the proposed algorithm has the highest CPU time, it outperformed the other algorithms in the remaining performance criteria.

To study the efficacy of the proposed algorithm, it is tested using two scenarios: increasing VMs while the number of tasks is constant or increasing the tasks while the number of VMs is constant. The performance of HHOLS is checked against another six metaheuristics using various performance measures

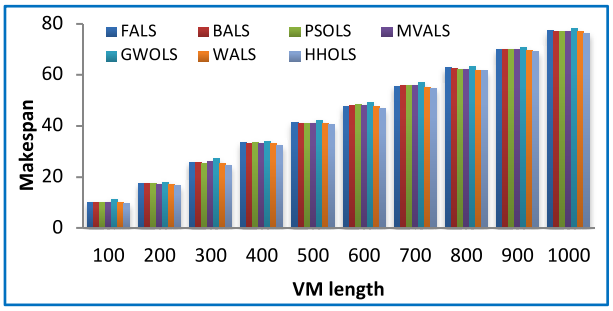

Fig. 12. Comparison based on the average makespan for different task lengths and 50 VMs.

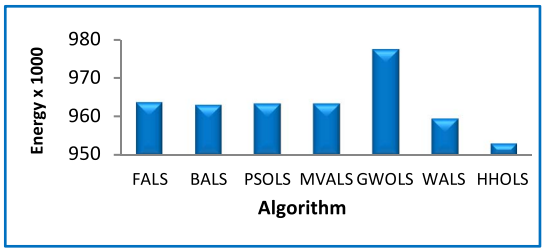

Fig. 13. Comparison based on the total energy in the average case for different task lengths and 50 VMs.

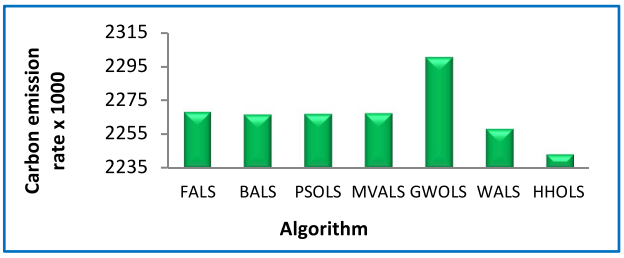

Fig. 14. Total CDERs in the average case for different task lengths and 50 VMs.

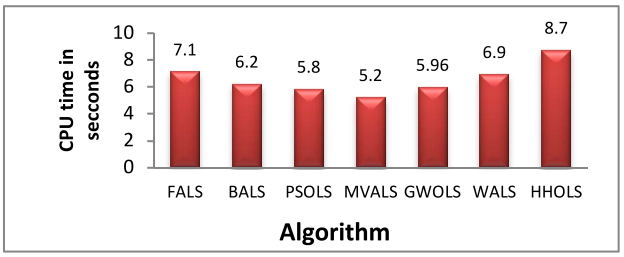

Fig. 15. Average CPU time of the algorithms.

(makespan, energy consumption, costs, flow time, CDER, and fitness). To ensure a fair comparison with all algorithms, they were improved using the same swap mutation and the local search strategy. Regarding the results shown above, we conclude that the HHOLS algorithm has proven its overwhelming success in achieving the lowest energy consumption and emission rates of carbon dioxide, as well as reducing the makespan and the costs of processing these tasks. The reason for the superiority of the proposed algorithm is its ability to balance between the exploitative and explorative capabilities.

VIII. Conclusion

In this article, a new energy-aware metaheuristic algorithm is proposed for the TSFC problem in IIoT applications. HHOLS is integrated with swap mutation operation and a local search strategy that improves its performance. The standard HHO algorithm is used for continuous problems; whereas task scheduling is a discrete problem. Accordingly, the normalization and scaling phase is applied for the continuous values obtained by the HHO algorithm. Several experiments are done to compare our proposed algorithm with other state-of-the-art algorithms (FALS, GWOLS, BALS, PSOLS, MVALS, and WALS) to verify HHOLS effectiveness. The candidate solutions can be evaluated using several performance measures: makespan, energy consumption, costs, flow time, and CDER. The energy consumption has a significant effect on costs and CDER. Also, makespan and flow time is important to determine the completion times of the tasks. Thus, a fitness function is suggested using the energy consumption and the makespan for evaluating the solutions. The fitness function is influenced by energy consumption more than the makespan as our main objective in this article is to maintain sustainable energy sources. The results show the outperformance of HHOLS. The exploitation capability of HHOLS is better than other algorithms. All this confirms that the proposed algorithm is financially and economically viable and conserving nonrenewable sources of energy that are difficult to obtain. Besides, the proposed algorithm has environmental benefits as it mitigates the CDER. Moreover, saving sustainable energy sources improves the performance of FC, so that the FC can face smart technologies, such as IoT requirements. Our proposed model is limited to schedule the independent tasks sent by the different users in the FC. Moreover, the local search strategy consumes much time. In the future, we need to apply our proposed algorithm to schedule the dependent tasks in the fog system. Also, we hope to extend our model to consider task migration. Furthermore, we hope to develop a parallel version of the proposed algorithm to make use of multiprocessing capabilities and reduce time consuming.

References

[1] CGC Index and CCVN Index, "Forecast and methodology 2016–2021," Cisco Syst., San Jose, CA, USA, White Paper, 2018.
[2] *Digitalization & Energy*, IEA Digit., Paris, France, 2017.
[3] A. A. Mutlag, M. K. A. Ghania, N. Arunkumar, M. A. Mohammed, and O. Mohd, "Enabling technologies for fog computing in healthcare IoT systems," *Future Gener. Comput. Syst.*, vol. 90, pp. 62–78, Jan. 2019.
[4] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Towards fault tolerant fog computing for IoT-based smart city applications," in *Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, USA, 2019, pp. 752–757.
[5] R. Contreras Masse, "Application of IoT with haptics interface in the smart manufacturing industry," *Instituto de Ingeniería y Tecnología*, vol. 10, no. 2, pp. 57–70, 2019.
[6] H.-C. Jang and T.-K. Lin, "Traffic-aware traffic signal control framework based on SDN and cloud-fog computing," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, Chicago, IL, USA, 2018, pp. 1–5.
[7] G. S. Aujla, N. Kumar, A. Y. Zomaya, and R. Ranjan, "Optimal decision making for big data processing at edge-cloud environment: An SDN perspective," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 778–789, Feb. 2018.
[8] A. Bouchentouf, *Commodities for Dummies*. Hoboken, NJ, USA: Wiley, 2011.

[9] B. K. Sovacool, "Valuing the greenhouse gas emissions from nuclear power: A critical survey," *Energy Policy*, vol. 36, no. 8, pp. 2950–2963, 2008.
[10] H.-Y. Wu and C.-R. Lee, "Energy efficient scheduling for heterogeneous fog computing architectures," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Tokyo, Japan, 2018, pp. 555–560.
[11] J. Li, J. Jin, D. Yuan, and H. Zhang, "Virtual fog: A virtualization enabled fog computing framework for Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 121–131, Feb. 2018.
[12] H. A. Khattak *et al.*, "Utilization and load balancing in fog servers for health applications," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 91, 2019.
[13] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *Proc. 18th Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, Kanazawa, Japan, 2016, pp. 1–4.
[14] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4712–4721, Oct. 2018.
[15] M.-Q. Tran, D. T. Nguyen, V. A. Le, D. H. Nguyen, and T. V. Pham, "Task placement on fog computing made efficient for IoT application provision," *Wireless Commun. Mobile Comput.*, vol. 2019, Jan. 2019, Art. no. 6215454.
[16] X. Xu, Q. Liu, L. Qi, Y. Yuan, W. Dou, and A. X. Liu, "A heuristic virtual machine scheduling method for load balancing in fog-cloud computing," in *Proc. IEEE 4th Int. Conf. Big Data Security Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput. (HPSC) Int. Conf. Intell. Data Security (IDS)*, Omaha, NE, USA, 2018, pp. 83–88.
[17] D. Rahbari and M. Nickray, "Low-latency and energy-efficient scheduling in fog-based IoT applications," *Turkish J. Elect. Eng. Comput. Sci.*, vol. 27, no. 2, pp. 1406–1427, 2019.
[18] H. Rafique, M. A. Shah, S. Ul 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.
[19] X. Yang and N. Rahmani, "Task scheduling mechanisms in fog computing: Review, trends, and perspectives," *Kybernetes*, to be published, doi: 10.1108/K-10-2019-0666.
[20] W.-C. Yeh, C.-M. Lai, and K.-C. Tseng, "Fog computing task scheduling optimization based on multi-objective simplified swarm optimization," *J. Phys. Conf. Series*, vol. 1411, no. 1, 2019, Art. no. 012007.
[21] B. M. Nguyen, H. Thi Thanh Binh, and B. Do Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," *Appl. Sci.*, vol. 9, no. 9, p. 1730, 2019.
[22] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Inf. Syst.*, vol. 12, no. 4, pp. 373–397, 2018.
[23] C. Guerrero, I. Lera, and C. Juiz, "Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures," *Future Gener. Comput. Syst.*, vol. 97, pp. 131–144, Aug. 2019.
[24] D. Rahbari and M. Nickray, "Scheduling of fog networks with optimized knapsack by symbiotic organisms search," in *Proc. 21st Conf. Open Innovat. Assoc. (FRUCT)*, Helsinki, Finland, 2017, pp. 278–283.
[25] 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 *Proc. Int. Conf. Comput. Inf. Sci. (ICCIS)*, Sakaka, Saudi Arabia, 2019, pp. 1–6.
[26] K. N. Jayasena and B. Thisarasinghe, "Optimized task scheduling on fog computing environment using meta heuristic algorithms," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Tokyo, Japan, 2019, pp. 53–58.
[27] 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," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, p. e3770, 2020.
[28] 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.
[29] P. Hosseinioun, M. Kheirabadi, S. R. K. Tabbakh, and R. Ghaemic, "A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm," *J. Parallel Distrib. Comput.*, vol. 143, pp. 88–96, Sep. 2020.
[30] S. K. Mishra, D. Puthal, J. J. P. C. Rodrigues, B. Sahoo, and E. Dutkiewicz, "Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4497–4506, Oct. 2018.
[31] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
[32] Y.-J. Zhang, Z. Liu, H. Zhang, and T. Tan, "The impact of economic growth, industrial structure and urbanization on carbon emission intensity in China," *Nat. Hazards*, vol. 73, no. 2, pp. 579–595, 2014.
[33] H.-J. Yang and J. He, "Analysis of the ethnic minority living activities CO_2 emissions from 2012 to 2014: A case study in Yunnan," in *Proc. Int. Conf. Adv. Mater. Energy Sustain. (AMES)*, 2017, pp. 611–620.
[34] M. Xie, Y. Bai, M. Huang, Y. Deng, and Z. Hu, "Energy-and-time-aware data acquisition for mobile robots using mixed cognition particle swarm optimization," *IEEE Internet Things J.*, early access, Apr. 29, 2020, doi: 10.1109/JIOT.2020.2991198.
[35] M. Mafarja, A. A. Heidari, M. Habib, H. Faris, T. Thaher, and I. Aljarah, "Augmented whale feature selection for IoT attacks: Structure, analysis and applications," *Future Gener. Comput. Syst.*, vol. 112, pp. 18–40, Nov. 2020.
[36] D. Yousri, M. Abd Elaziz, and S. Mirjalili, "Fractional-order calculus-based flower pollination algorithm with local search for global optimization and image segmentation," *Knowl. Based Syst.*, vol. 197, Jun. 2020, Art. no. 105889.
[37] B. Cao, S. Zhao, X. Li, and B. Wang, "K-means multi-verse optimizer (KMVO) algorithm to construct DNA storage codes," *IEEE Access*, vol. 8, pp. 29547–29556, 2020.
[38] M. Guo, J. Wang, L. Zhu, S. Guo, and W. Xie, "An improved grey wolf optimizer based on tracking and seeking modes to solve function optimization problems," *IEEE Access*, vol. 8, pp. 69861–69893, 2020.
[39] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Proc. Int. Conf. Unconv. Comput. Nat. Comput.*, 2012, pp. 240–249.
[40] J. Kennedy, *Particle Swarm Optimization. Encyclopedia of Machine Learning*, vol. 10. Boston, MA, USA: Springer, 2010, pp. 760–766.
[41] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," *Neural Comput. Appl.*, vol. 27, no. 2, pp. 495–513, 2016.
[42] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
[43] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.
[44] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. Heidelberg, Germany: Springer, 2010, pp. 65–74.

Mohamed Abdel-Basset (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in operations research from the Faculty of Computers and Informatics, Zagazig University, Zagazig, Egypt.

He is an Associate Professor with the Department of Computer Science, Zagazig University. He is working on the application of multiobjective and robust meta-heuristic optimization techniques. He has published more than 170 articles in international journals and conference proceedings. His current research interests are optimization, data mining, computational intelligence, robust optimization, engineering optimization, multiobjective optimization, swarm intelligence, evolutionary algorithms, and soft computing.

Dr. Abdel-Basset is an editor/reviewer for different international journals and conferences.

Doaa El-Shahat received the B.Sc. degree from the Faculty of Computers and Informatics, Department of Computer Science, Zagazig University, Zagazig, Egypt.

Her area of interests include computation intelligence, optimization, swarm intelligence, evolutionary algorithms, and artificial neural networks.

Mohamed Elhoseny (Senior Member, IEEE) received the Ph.D. degrees in computers and information from Mansoura University, Mansoura, Egypt, and the University of North Texas, Denton, TX, USA, through a joint scientific program.

He is an Assistant Professor with the Faculty of Computers and Information, Mansoura University.

Dr. Elhoseny is the Founder and the Editor-in-Chief of the *International Journal of Smart Sensor Technologies and Applications* (IGI Global). He is an Associate Editor of the IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, IEEE ACCESS, *Scientific Reports*, IEEE Future Directions, *Remote Sensing*, and the *International Journal of e-Services and Mobile Applications*. He is the Editor-in-Chief of *Studies in Distributed Intelligence* (Springer), *Sensors Communication for Urban Intelligence* (CRC Press-Taylor& Francis), and *Distributed Sensing and Intelligent Systems* (CRC Press-Taylor& Francis). He is an ACM Distinguished Speaker.

Houbing Song (Senior Member, IEEE) received the M.S. degree in civil engineering from the University of Texas, El Paso, TX, USA, in December 2006, and the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in August 2012.

In August 2017, he joined the Department of Electrical Engineering and Computer Science, Embry–Riddle Aeronautical University, Daytona Beach, FL, USA, where he is currently an Assistant Professor and the Director with the Security and Optimization for Networked Globe Laboratory. He served on the Faculty with West Virginia University, Morgantown, WV, USA, from August 2012 to August 2017. In 2007, he was an Engineering Research Associate with Texas A&M Transportation Institute, College Station, TX, USA.

Dr. Song has served as an Associate Technical Editor for *IEEE Communications Magazine* since 2017, an Associate Editor for the IEEE INTERNET OF THINGS JOURNAL since 2020, and the IEEE JOURNAL ON MINIATURIZATION FOR AIR AND SPACE SYSTEMS since 2020.