IETIF: Intelligent Energy-aware Task Scheduling Technique in IoT/Fog Networks

Amin Nazari
Bu-Ali Sina University
Sakine Sohrabi
Bu-Ali Sina University
Reza Mohammadi (r.mohammadi@basu.ac.ir)
Bu-Ali Sina University
Mohammad Nasiri
Bu-Ali Sina University
Muharram Mansoorizadeh
Bu-Ali Sina University

Research Article

Keywords: Task scheduling, Fog Computing, Internet of Things, Energy Efficiency

Posted Date: March 21st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1454775/v1

Additional Declarations: No competing interests reported.

IETIF: Intelligent Energy-aware Task Scheduling Technique in IoT/Fog Networks

Amin Nazari, Sakine Sohrabi, *Reza Mohammadi, Mohammad Nasiri, Muharram Mansoorizadeh

Department of Computer Engineering, Engineering Faculty, Bu-Ali Sina University, Hamedan, Iran.

Abstract: Nowadays, with the advent of various communication technologies such as Internet of Things (IoT), a large volume of data is produced and needs to be processed in real-time. Fog computing is an appropriate solution to address the requirements of different types of IoT applications. In most cases, IoT applications consists a set of dependent task which can be separately processed in the heterogeneous Fog environment. Scheduling such dependent tasks in Fog environment is an NP-hard problem which needs a long time to be solved,making it unsuitable for real-time applications. In addition, reducing response time and energy consumption in Fog computing is an essential issue which should be taken into account in task scheduling algorithm. To address these challenges, we aim to propose a multiobjective task scheduling model to jointly improve energy efficiency and response time. To solve the model, we also propose an intelligent solution named IETIF which combines and levergaes the benefits of simulated annealing and NSGA-III algorithms. Simulation results show that IETIF outperforms the other state-of-the-art methods in terms of energy, response time and speedup.

Keywords: Task scheduling, Fog Computing, Internet of Things, Energy Efficiency.

1- Introduction

Emerging technologies, such as the Internet of Things, have led to massive data production. The use of traditional and standard methods for processing various forms of data has been challenged by the abrupt increase in the volume of data generated. [1]. Many applications in the Internet of Things generate a great deal of data through a wide variety of sensors, necessitating latency-aware computations for real time processing. [2, 3].

Cloud computing is built on a large computer networks which could be helpful in terms of information security, fast processing, dynamic access and cost saving. [4]. It's inefficient to process IoT requests exclusively over the cloud , especially for time-critical IoT applications. As a result of delays in data transfer and processing in the cloud layer, system performance is reduced. Fog computing, which acts as an intermediate layer between the cloud layer and IoT devices, has been proposed as a solution to this problem. [5].

Bonomi et al. [6] proposed the computational model as a complete virtual platform and was the first definition of fog computing. IoT devices link to fog computing devices in the fog computing environment. These devices are close to the users and are responsible for calculating and storing data in the middle layer. This solution minimizes system latency and increases speed. Resource allocation and scheduling are among the most challenging aspects of executing IoT applications in the fog computing environment [2].

The fog environment's heterogeneity of resources, dynamic nature, and resource limitations make task scheduling challenging. Since task scheduling is a complete NP problem, finding optimal solutions necessitates novel approaches [7-9]. Fog computations are still in their infancy, and resource allocation and job scheduling concerns are essential challenges. Because fog devices are in the vicinity of the network and they have limited resources [10]. A scheduling algorithm is a mechanism for allocating tasks to system resources. It should be noted that the incompatible scheduling algorithm might cause hardware inefficiencies or slow down the applications. But, using the compatible algorithm reduces Energy consumption and response time. [11].

A method for scheduling jobs in the fog environment is presented in this research. A genetic algorithm with Non-dominated sorting and a Simulated Annealing algorithm is used in the suggested method. The research's primary innovations and points are described below.

1. Get priority tasks in the form of a DAG graph, with graph creation and weighting dependent on

- Corresponding Author: Reza Mohammadi (r.mohammadi@basu.ac.ir)

network communication and transmission delay, and assign priority jobs dynamically.
2. Create clever first solutions for the proposed algorithm.
3. Consider the impact of communication delays on system reaction times.
4. using a multiobjective algorithm to optimize energy consumption and response time simultaneously
5. Using the DVFS method, select the best choice from the list of alternatives.

Simulation results demonstrate the suggested approach outperforms Random HEFT, HEFT and IWOCA algorithm in some criteria. The paper is organized as follows: The second section gives an outline of the study's findings. The model of scheduling and problem formulation is explained in the third section. The fourth section of this study describes the methods employed in the proposed algorithm, which is based on the genetic algorithm with Non-dominated sorting and the simulated annealing algorithm. IETIF is described in the fifth part. The findings of experiments and simulations are presented in the sixth section, and the algorithm's results are examined in the seventh section.

2- Review

Because of its computations near the network's edge and proximity to the user, the fog computing environment has attracted a lot of interest. Due to resource constraints, IoT node mobility, and service quality limitations, scheduling has become a problematic challenge for distributing resources correctly in this environment. Intelligent transportation, emergency, online gaming, telecommunications, and digital signal processing are high-performance, low-latency applications. These applications require quick processing, low latency, and incredible responsiveness and dependability, resulting in computational overhead, increased communication, and network congestion. Therefore, we need a suitable scheduling algorithm for resource allocation in the fog environment.

Many researchers have looked at the challenge of job scheduling in heterogeneous fog-cloud computing systems. In general, they aim to reduce program execution time and some of them also consider other factors such as energy consumption, Reliability, and so on.

We will discuss some of the research works that it has been done on the subject of task scheduling in the fog environment in the following sections. We investigate the research gaps in this field and offer the suggested method by comparing the presented works.

Li et al. presented a combination of two time-aware and energy-aware algorithms to task scheduling in heterogeneous computational systems. According to the requirements of the consumers, the algorithm reduces energy consumption or saves time. It should be mentioned that there are still some issues with load balancing in this method [11]. Another approach is the ASSD algorithm, which considers the time limit and energy consumption of resources under various voltages to accelerate parallel tasks. This method considers two sorts of deadlines for completing the work: soft and harsh deadlines, and it is also based on a dynamic voltage and frequency calculation model (DVFS). This method considers the system workload and then determines the voltage and frequency based on the system workload. The results of the simulations reveal that not only energy are saved but also the number of performing tasks are enhanced [12]. MDAF algorithm is an efficient and unique algorithm for scheduling and assigning services based on the QoScriteria. The algorithm performs more time-sensitive operations at the layer closer to the client than possible. The simulation results reveal that the algorithm has dramatically improved service delays and implementation costs. Still, energy efficiency and resource utilization have not been addressed, and resources are not being used properly [13]. Shojaeifar et al. developed an approach that reduces latency and makes the best use of fog resources. Although the suggested SJF method utilized in this paper minimizes average waiting time and network usage, it suffers from scheduling starvation [10].

The MinRE algorithm is a scheduling policy developed for allocating services according to QoS criteria such as response time, power consumption, and resource utilization. In this strategy, services are divided into two categories: critical and non-critical, and the MinRes method is utilized for critical services to reduce response time, At the same time, the MinEng algorithm is recommended for non-critical services. Its goal is to lower energy consumption in the fog environment. It should be emphasized that the cost of resources and interdependence between numerous services are not considered in this method [8].

The JointPT algorithm is a service scheduling and scheduling policy based on priority, power, and traffic indicators. The fat-tree topology is employed in this method, and an analysis of the findings demonstrates that the proposed method can reduce total network usage, energy consumption, and resource waste. Today, researchers are developing recommended solutions for application in the IoT-Fog-Cloud ecosystem [14]. FPFTS algorithm

combines PSO and fuzzy approaches. As input parameters, the algorithm examines the capacity to compute resources and user job needs. This method presents a service allocation policy based on latency and effective network resource utilization. Their method consideres a mobility-aware strategy, but the priority of implementation has not been taken into account [15]. Wang et al. created I-FASC algorithm. I-FA fireworks algorithm, a cumulative intelligence optimization technique, is used. The simulation findings show that convergence speed, workload balance, and completion time are improved. It should be emphasized that there are still many flaws in the actual program's development. The energy consumption of fog nodes, for example, is not taken into consideration when calculating work processing time and workload optimization [16]. LBP-ACS algorithm is a combination of LBPA and ACS algorithms proposed to solve the problem of scheduling work in the fog computing environment. In this algorithm, the priority of a task and its expiration date is considered. Also, to address the sensitivity of work delay, the axity-based priority algorithm is used to schedule and build a sequence of work with reasonable priority. In order to minimize energy consumption, an ant colony optimization algorithm is used. There is no optimal answer to scheduling using the above method for interdependent work [17]. TCaS algorithm, based on the GA algorithm, is a work scheduling method. The cost and time of tasks are among the criteria covered by this strategy. It's worth noting that the criteria of transfer cost, energy consumption, and user satisfaction aren't addressed in this method [18]. IPSO algorithm, based on the PSO algorithm, is a task scheduling approach. The tasks are scheduled with the goal of decreasing the cost and makespan. The simulation results suggest that the proposed scheduling algorithm is more cost-effective than the primary method and can outperform it [19].

3- System Model

First, we'll go over the desired system architecture in this section. The problem formulation is offered after that.

3-1- System Architecture

The suggested system architecture comprises three layers: Cloud data centers on the top layer, Heterogeneous nodes in a fog layer, and an end-user layer of IoT devices. The system architecture is depicted in Figure (1) at the bottom layer.

Fig.1: system architecture

Figure (1) Describes how requests from IoT devices are sent to the fog layer as a task for processing (second layer). Each task can be broken down into several smaller tasks that are either interdependent or independent. The broker is the most essential component of the fog layer. It receives requests and manages them based on information about processing resources such as processing cost, computational power, and other scheduling algorithm characteristics.eventually, Executes requests that have been sent to end-users.

It should be noted that in some situations, user requests for processing are routed to the cloud layer, which also includes a set of virtual machines with significantly more computing power than the fog layer's virtual machines. However, this results in a longer transfer time.

The edges in the DAG graph examine the workflow and prioritize the tasks. An entry task has no priorityand an exit task has no successor. Tasks and edges have weights in this graph, used to determine computational and communication costs. A simple DAG graph is shown in figure (1). There are 11 tasks and workflows demonstrated between them. The graph shows that t_0 represents the input task and t_{10} represents the output task. Also, if t_0 and t_1 are both running on the same host, the transfer cost (or transfer time) will be zero, but if they are both running on separate hosts, the transfer cost (or transfer time) will be 3. The majority of previous algorithms focus on lowering task execution time. In contrast, we consider the measures of response time(which includes execution time and transfer time) and energy in this article.

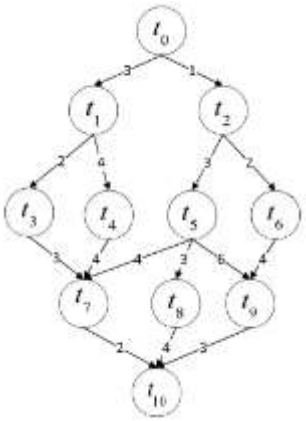

Fig.2: DAG graph

3-2- Problem Formulation

The suggested method strives to reduce response time and energy consumption at the same time, and we'll show you how to calculate them in the next section.

3-2-1- Response Time

Equation (1) is used to calculate response time.

$$\boldsymbol{ResponseTime = Makespan + NetworkDelay} \quad (1)$$

The execution time in this technique, corresponds to the makespan of the last job, and its value is calculated using equation (2).

$$\boldsymbol{Makepan = min\,(max\,(EFT)) = max\,(AFT)} \quad (2)$$

The EFT value represents the closest completion time, which is computed by equation (3).

$$\boldsymbol{EFT(t_i, h_k) = t_{ij} + AST(t_i, h_k)} \quad (3)$$

The variable t_{ij} indicates the amount of time the ith task is run on the jth hosts, and the variable $AST(t_i, h_k)$ shows the actual start time of the ith task on the jth hosts. The AFT variable's value also indicates the actual completion time, determined using equation (4).

$$\boldsymbol{AFT(t_i, h_k) = min_{1 \le k \le H} EFT(t_i, h_k)} \quad (4)$$

3-2-2- Transmission Delay

The transmission delay is predetermined and determined by the DAG graph in prior work. In contrast, in this method, the transmission delay is determined by the network structure and the source and destination locations. The network topology, transmission costs and network equipment energy consumption in the fog layer are considered in figure 3.

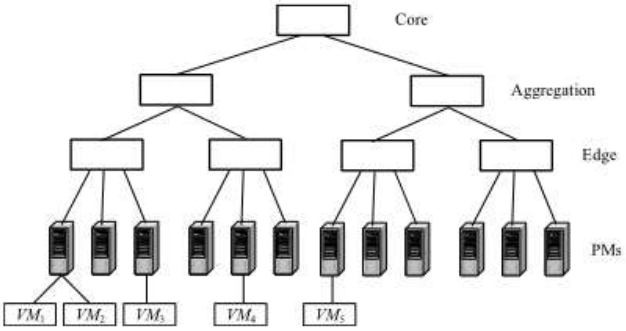

Fig.3: Communication network

The cost or delay of transmission is determined using the suggested technique based on the three-level network topology (dynamically) and each solution utilizes equation (5). If the t_i task needs to migrate, the transfer delay is obtained from the following relation, otherwise its value is equal to zero.

$$\boldsymbol{D_T(t_i) = \begin{cases} d_t * l_t \\ 0 \end{cases}} \quad (5)$$

Where l_t denotes the size of the task and d_t is determined based on the distance between the source and destination machines. The value d_t will be equal to d_1 if the two physical machines of origin and destination are both linked to an edge switch. If the link between two devices is formed using an aggregation switch, d_t will be equal to d_2 . The delay is d_3 if the core switch must pass for transmission. Equation (6) explains calculating d_t between two i and j machines.

$$\boldsymbol{d_t = \begin{cases} d_1 & j,k \text{ are connected to same edge SW} \\ d_2 & j,k \text{ are connected by aggregation SW} \\ d_3 & j,k \text{ are connected core SW} \end{cases}} \quad (6)$$

The amount of network delay is shown in equation (7).

$$\boldsymbol{NetworkDelay = \sum_{i=1}^{H} D_T(t_i)} \quad (7)$$

3-2-3- Energy Consumption

Energy consumption in cloud computing is calculated according to CPU efficiency using equation (8) (Ye, Yanli, and Lan, 2017).

$$\boldsymbol{E_C(h_j) = \begin{cases} \{E_{idel} + (E_{busy} - E_{idel}) * U_j^{cpu}, U_j^{cpu} > 0 \\ 0 \quad , U_j^{cpu} = 0 \end{cases}} \quad (8)$$

$E(h_j)$ tells how much energy the machine h_j consumes. The value of the variable E_{idel} denotes the

amount of energy consumed if the physical machine is turned on or idle. The value of the variable E_{busy} represents the amount of energy spent by the host at its most efficient status. In contrast, the value of the variable U_j^{cpu} represents the machine's efficiency, as estimated by equation (9).

$$U_j^{cpu} = \begin{cases} \frac{t_i^{cpu}}{h_j^{cpu}} \\ 0 \end{cases} \quad (9)$$

If task i is done on machine j, U_j^{cpu} represents the ratio of the MIPS required for task i to the CPU capacity of machine j's. We modify the utility formula as equation (10), intending to consider the task time and frequency of the processor.

$$U_j^{cpu} = \sum_{i \epsilon A(j)} PT_i * VF_j \quad (10)$$

Where PT demonstrates the processing time of the job i on the machine j and VF indicates the amount of voltage consumed by the CPU at frequency of f. The overall energy cost for task migration is calculated by $\sum_{t_i \in M} E_T(t_i)$. The migration cost of the t_i job is computed using equation (11) if migration is required; otherwise, the migration cost is zero.

$$E_T(t_i) = \begin{cases} e_t * l_t \\ 0 \end{cases} \quad (11)$$

l_t is the size of the task and e_t is determined by using equation (12) based on the location of the two hosts.

$$e_t = \begin{cases} e_1 & j,k \text{ are connected to same edge SW} \\ e_2 & j,k \text{ are connected by aggregation SW} \\ e_3 & j,k \text{ are connected core SW} \end{cases} \quad (12)$$

Therefore, energy consumption is calculated using equation (13).

$$E_{total} = \sum_{j=1}^{H} E_C(h_j) + \sum_{t_i \in M} E_T(t_i) \quad (13)$$

3-2-4- Fitness function

As aforementioned, the suggested method is a multiobjective optimization problem, and any of the solutions in Pareto-front could be the answer. A difficulty is selecting the best solution from your existing options. We will use equation (14) to normalize and weigh each goal, then choose the best response.

$$min(\alpha * Normalize(ResponseTime) + \beta * Normalize(Energy)) \quad (14)$$

The significance coefficients of each aim are represented by the values of the variables α and β. In addition, the data is standardized with the use of the normalize function in equation (15) to account for the influence of each aim equally.

$$Normalize(x, x_i) = abs(mean(x) - x_i)/std(x) \quad (15)$$

4- NSGA-III and SA algorithms

IETIF uses a combination of NSGA-III and SA multiobjective algorithms and improve its operators to solve scheduling problems, each of which we will briefly explain below.

4-1- NSGA-III algorithm

To solve multiobjective problems, various evolutionary algorithms have been developed, including the genetic algorithm with uncoordinated sorting (NSGA-III). The NSGA-II [20]. NSGA-III algorithm employs a fixed number of reference points. The use of these reference points has led to the well-being of population expansion at Pareto-front and the diversity of solutions.

There are usually numerous separate objective functions that tend to find their maximum or minimum at the same time in multiobjective optimization problems. It's worth noting that the majority of these goal functions are at opposite points to each other. So, improving one of them leads to worsening the other [21]. Therefore, a set of optimal answers is obtained which is called the Pareto front. The collection of selected solutions on the Pareto front are not dominated by any other solution and are scattered by reference points in the problem space. Experts determine the number of reference points. The reference points of a problem with a multiobjective function are located on a normalized superplane with m-1 dimension. In this way, each dimension is partitioned into p equal sections. Equation (16) is used to calculate the total number of reference points.

$$H = \binom{m + p - 1}{p} \quad (16)$$

4-2- Simulated Annealing Algorithm

The Simulated Annealing (SA) method starts with an initial solution and then proceeds to surround solutions in an iterative loop when addressing an optimization issue. The algorithm sets the neighbor's response as the current answer (moves to it) if it is better than the current answer; otherwise, it accepts

that answer as the current solution with exp(-ΔE/T) probability. ΔE is the difference between the objective function of the present response, and the nearby answer in this relationship, and T is a temperature parameter. At each temperature, several repetitions are performed and then the temperature is slowly reduced.
In the initial step the temperature is set too high to be more likely to accept worse solutions (simulating the free movement of molecules at high temperatures). In the final steps there is less chance of accepting worse answers, and so the algorithm converges towards a global optimal. The SA algorithm's most significant benefit is that it avoids local optimizations. Accepting incorrect answers raises the chances of finding a solution in the problem space.

5- IETIF Architecture

IETIF is a hybrid multiobjective algorithm that consists of three main phases, which we will examine in the following.

5-1- Primary population production

The first stage is to create the initial population after specifying the required parameters. IETIF replaces the generation of the initial population by a random approach with producing the production in an intelligent way. As a result, each particle is regarded as a possible solution. Each particle is represented by an array along the number of tasks. Tasks must be completed in the order in which they appear in the array. In other words, the index of the array represents the task's priority and the value of the element shows the task number. Figure 4 is the first attempt at solving task scheduling in figure 2.

q	0	1	2	3	4	5	6	7	8	9	10	11

Fig.4: A sample solution

The solutions developed in the initial population must meet these limits because the workflow is DAG-based, and the sequence of execution is critical. The node height values in the DAG are used to construct the starting population in this approach.
The minimum height of each DAG node is calculated using equation (17).

$$h(t_i) = \begin{cases} 0 & pred(t_i) = \emptyset \\ 1 + max_{t_{j \in pred(t_j)}}(h(t_j)) & pred(t_i) \neq \emptyset \end{cases} \quad (17)$$

We use three operators to construct the initial population: level arrangement, deep arrangement, and hybrid.

5-1-1- Level arrangement

The jobs are sorted in ascending order depending on their height value in this scenario. The task with the smallest height is at the top of the list, while the task with the largest height is at the bottom. The tasks with the same height are picked and a permutation of them include in the list based on the height (from less to more). This is repeated until all of the tasks have been completed. This operator ensures that tasks at lower heights is always completed sooner than tasks at higher heights. It is similar to a breadth-first search in that each run yields a different result.

5-1-2- Deep arrangement

The job arrangement in this scenario is not always dependent on height. In figure 1, for example, Possibly, the fourth task could be accomplished before the second one. The level arrangement operator will not create this state. So, we present the deep arrangement operator to examine the problem space in more depth. This operator works as a depth-first search, returning a different but correct result each time used. The incoming work is initially chosen, and all of its children are added to the candidate list. Then, one of the nodes of the candidate list is selected randomly. Among the children of the selected node, nodes that have no dependence on nodes that have not yet been seen are candidates for selection in the next step. This process continues until all nodes are listed.

5-1-3- combined

This mode combines deep and level search and generates solutions using both methods. Operators can be used at equal or random rates to generate the initial population.

5-2- NSGA-iii and SA combined algorithm

To produce new solutions, we'll use the simulated annealing algorithm's operators in addition to the mutation operator. As a result, the new solution after performing each of these operators is an executable solution.

5-2-1- Reversion operator

In this situation, a task is chosen randomly, followed by all tasks at the same level. In the new solution, the order of the selected tasks is reversed. Assume that q is a proposed solution and task 4 is chosen. Because the third, fourth, fifth, and sixth tasks are all on the same level. The sequence in which they appear on the new list has been switched around. Figure 5 shows the output of this operator.

q	0	1	2	3	4	5	6	7	8	9	10	11
qnew	0	1	2	6	5	4	3	7	8	9	10	**11**

Fig.5: Result of the reversion operator

5-2-2- Insertion operator

In this strategy, a task is chosen randomly first i_1. The second task is selected between all tasks that are at the same height as i_1. Then a random number r between zero and one is used to move. If $r \geq 0.5$, i_2 is ahead of i_1. If the value of $r < 0.5$, i_1 moves after i_2. If q is a solution and tasks 3 and 6 are chosen randomly, the insert operator in figure 6 is set for $r < 0.5$ and $r \geq 0.5$.

q	0	1	2	3	4	5	6	7	8	9	10	11
qnew	0	1	2	6	3	4	5	7	8	9	10	11
qnew	0	1	2	4	5	6	3	7	8	9	10	11

Fig.6: Insert operator result

5-2-3- Replacement operator

In this strategy, a task is chosen randomly first i_1. Then all of the tasks that are at the same height as the selected task are chosen, and a second task is chosen from i_2. The element now in position i_1 will be moved to position i_2, and the element currently in position i_2 will be transferred to position i_1. Assume that positions 7 and 9 are chosen. Figure 7 shows the result after the moving procedure.

q	0	1	2	3	4	5	6	7	8	9	10	11
qnew	0	1	2	3	4	5	6	9	8	7	10	11

Fig.7: Result of the displacement operator

5-2-4- Mutation operator

In this procedure, a node is chosen randomly first i_1. The parent nodes and the children of node i_1 are identified. The i_1 node is then moved after the final parent or before the first child using a random number.

5-3- DVFS

The usage of the dynamic voltage and frequency scale (DVFS) technique is one of the most important and extensively utilized dynamic ways for controlling the heat generated by circuits and reducing energy consumption in microprocessors. The processor's voltage and frequency are changed to reduce power usage in this technology dynamically. The DVFS algorithm can dramatically cut energy usage. This algorithm's primary purpose is to adjust the voltage and frequency of existing CPUs in real-time. The primary goal of this technique is to figure out what voltage and frequency the processors should run at during program execution, which saves energy. For this purpose, we adjust the execution frequency in ascending order for each task and update the completion time to identify the lowest frequency that will allow the program to be completed due to the time limitations [22, 23]. The best solution available in the Pareto-front was chosen using the DVFS approach. IETIF pseudo-code is shown in figure 8.

Algorithm.1: IETIF Algorithm

Input: Tasks, Vms, Cost Of Computations and Cost of Communications

Output: The Optimal (Sub-Optimal) Tasks Assignment

```
1:t ← 0
2:For i=1 to npop
  Begin:
3:    r = rand();
4:    if (r< 1/3) then
5:        p(i)^t ← Level Assignment;
6:    elseif (1/3 ≤r< < 2/3) then
7:        p(i)^t ← Deep Assignment;
8:    else
9:        p(i)^t ← Combined Assignment;
      End if
  End for
10:Evaluation Population;
11:while (t < MaxGeneration)
  Begin:
12:   For i=1 to npop
13:       r = rand();
14:       if (r< 1/4) then
15:           p(i)^(t+1) ← Reversion(p(i)^t);
16:       elseif (1/4 ≤r< < 1/2) then
17:           p(i)^(t+1) ← Insetion(p(i)^t);
18:       elseif (1/2 ≤r< < 3/4) then
19:           p(i)^(t+1) ← Replacment(p(i)^t);
20:       else
21:           p(i)^(t+1) ← Mutation(p(i)^t);
22:       End if
      End for
23:   t ← t + 1
  End while
24:Return Best Solution from Pareto-front;
```

Fig.8: IETIF Algorithm

6- Evaluation

In this section, the performance of IETIF is evaluated in terms of energy consumption, response time, speedup, efficiency, and SLR. IETIF will be compared to recent studies such as HEFT-U, HEFT-D, HEFT-L [24], RandomHEFT [25], and IWO-CA [23]. Random DAGs are created in experiments to examine the performance of algorithms under various scenarios. An Intel Core i7 CPU and 16 GB of memory was utilized to develop MATLAB algorithms. The simulation conditions are given in table 1.

Table 1. Conditions for simulation using the studied algorithms

	IWO_CA	NSGAIII SA
Max – iteration	100	100
Number of population	100	100
Number of reference points	-	10
Number of seeds	(0,5)	-

Equations 1 and 13 describe the response time and energy consumption criteria. Other parameters will be presented in the following.

6-1- Random DAGs

Random DAGs with 10, 20, 40 and 80 tasks and height $\theta(\log(n))$ are created. The communication to computation ratio (CCR) parameter creates distinct datasets DAGs to analyze the proposed method in various scenarios. We make two datasets, with CCR one and two. In addition, the produced programs include a variety of computations and communication expenses generated randomly. Dataset properties are listed in table 2.

Table 2. Dataset features

		Num. tasks	Num. hosts	Comp. Cost	Comm. Cost
Scenario 1	CCR=1	10	3	5-15	10-41
	CCR=2				22-78
Scenario 2	CCR=1	20	5	5-15	10-99
	CCR=2				20-129
Scenario 3	CCR=1	40	10	5-15	20-94
	CCR=2				15-162
Scenario 4	CCR=1	80	15	5-15	10-96
	CCR=2				13-135

6-1- Speedup

The ratio of sequential execution time for parallel execution time is known as speedup. The sequential execution time is calculated by allocating all tasks to the processor that minimizes the task graph's total execution time, as shown in equation (18).

$$\boldsymbol{Speedup} = \frac{\boldsymbol{min_{p_j \epsilon P}[\sum_{t_i \epsilon T} C_{comp}(t_i, p_j)]}}{\boldsymbol{makespan}} \qquad (18)$$

6-2- Effieciency

Efficiency is defined as the ratio of the speedup value to the number of processors utilized. Equation (19) is used to calculate this measure. The efficiency measure displays how many virtual machines are being used to boost performance.

$$\boldsymbol{efficiency} = \frac{\boldsymbol{speedup}}{\boldsymbol{number\ of\ used\ VM}} * \mathbf{100\%} \qquad (19)$$

6-3- Scheduling length ratio (SLR)

The makespan is the most often used metric for comparing the performance of scheduling algorithms. However, we use the normalized schedule length (NSL), also known as the scheduling length ratio, to normalize the schedule length against any topology/processor graph (SLR).

$$\boldsymbol{SLR} = \frac{\boldsymbol{makespan}}{\sum_{t_i \epsilon CP} \boldsymbol{min_{p_j \epsilon P}[C_{comp}(t_i, p_j)]}} \qquad (20)$$

Where CP is the critical path, the critical path is the longest path from the entry node to the exit node in the application graph. $C_{comp}(t_i, p_j)$ Shows the computational cost of the ith task on the jth host. Since the critical path serves as a lower bound for the makespan, the SLR cannot be less than one. The denominator value may be smaller than the actual critical path because this formulation ignores communication costs and thus produces shorter critical path lengths than the valid critical path length.

6-4- Results and analysis

All algorithms were run ten times to evaluate and compare the results, and the average is reported in this section. Table 3 illustrates the percentage improvement of the suggested approach over other methods for studied criteria.

Table 3. percentage improvement

		IWO-CA	HEFT-Upward	HEFT-Downward	HEFT-Level	RandomHEFT
Energy	CCR=1	5.81	1.67	3.67	2.71	-18.68
	CCR=2	8.42	3.59	4.96	2.27	-16.51
Response Time	CCR=1	1.11	9.32	13.41	9.78	26.95
	CCR=2	1.46	2.26	12.91	8.94	22.31
SLR	CCR=1	1.15	10.31	14.3	9.88	27.36
	CCR=2	1.34	2.61	12.83	9.17	22.79
Efficiency	CCR=1	1.64	22.49	23	9.79	39.08

	CCR=2	0.56	7.48	13.19	11.98	38.41
Speedup	CCR=1	1.12	12.76	16.3	9.98	36.13
	CCR=2	1.28	3.35	14.56	10.16	30.09

One of the most important criteria that has been considered in most articles is response time. Response time criteria and energy are considered simultaneously thanks to a multiobjective algorithm, which has improved the performance of IETIF. The results for CCR =1 and CCR=2 are shown in figures 9 and 10.

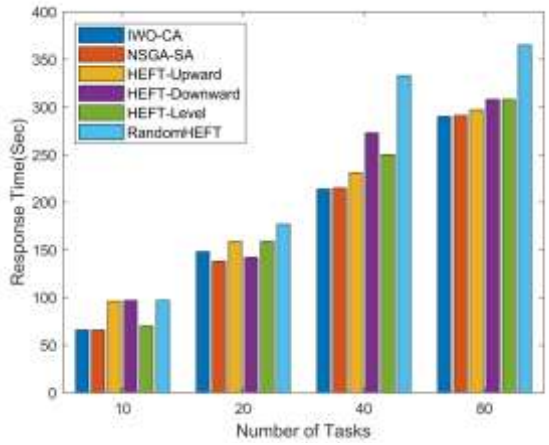

Fig.9: Response time(CCR=1)

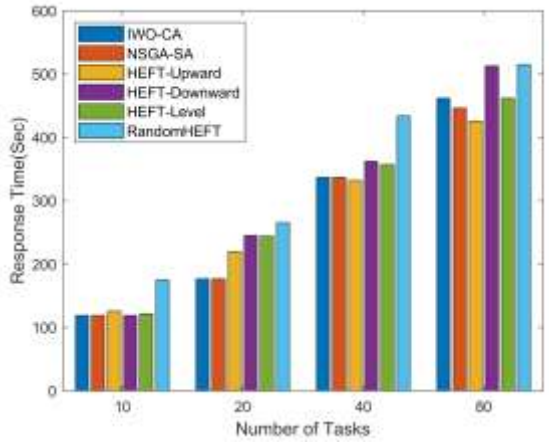

Fig.10: Response time(CCR=2)

Figures 9 and 10 illustrate that our approach improves response time compared to other algorithms. in addition, By increasing the number of tasks entered into the system, response time of IETIF is significantly decreased, resulting in enhanced system performance. Table 3 shows that when the CCR=1, response time for IETIF has improved by 13.41% and 26.95% , respectively, compared to the HEFT-D and RandomHEFT methods. When the CCR=2, the percentage of improvement will be 12.91% and 22.31%, respectively. The energy consumption criteria for all methods are shown in figures 11 and 12.

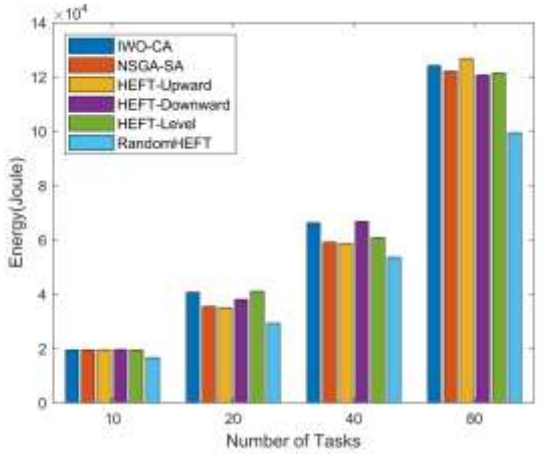

Fig.11: Energy Consumption(CCR=1)

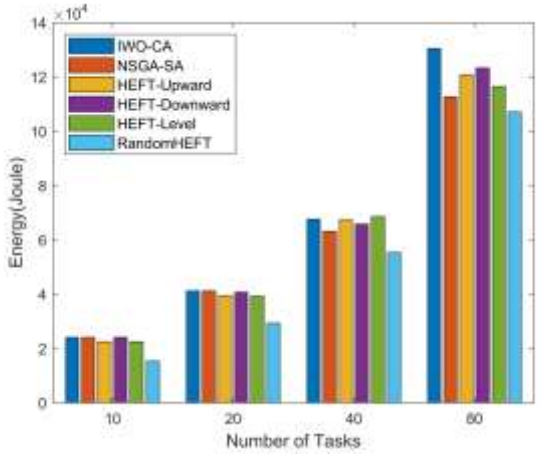

Fig.12: Energy Consumption(CCR=2)

By comparing the simulation results, it is clear that IETIF outperforms other approaches. There is a substantial difference in the amount of energy required by the suggested technique compared to other algorithms. The number of tasks submitted into the system increases, and IETIF is superior to other methods in terms of energy consumption. When CCR = 1, for example, the suggested method reduces energy consumption by 5.81% and 3.67%, respectively, compared to IWO-CA and HEFT-D, as shown in Table 3 When CCR=2, the percentage of improvement in energy usage is more excellent, this rate rises. Figures 13 and 14 show the results of speedup for various scenarios.

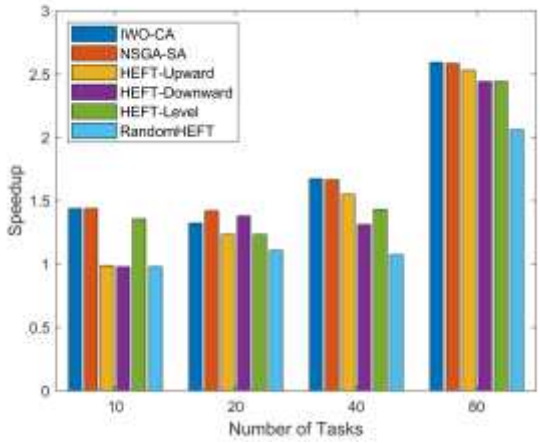

Fig.13: Speedup (CCR=1)

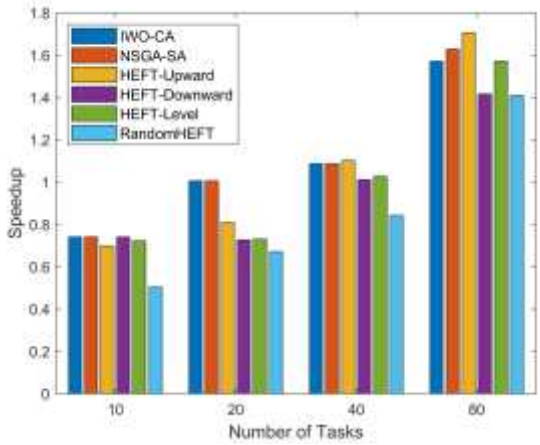

Fig.14: Speedup(CCR=2)

The speedup parameter specifies how quickly tasks are completed, when they enter into the system. According to the results shown in figures 13 and 14, our technique is faster than other algorithms. It has provided much better results. for example, when the CCR=1, as shown in table (3) our method enhances the speedup parameter by 16.3% and 36.13%, respectively, compared to the HEFT-D and RandomHEFT methods. When CCR=2, this rate will be 14.56% and 30.09%, respectively. Figures 15 and 16 demonstrate efficiency.

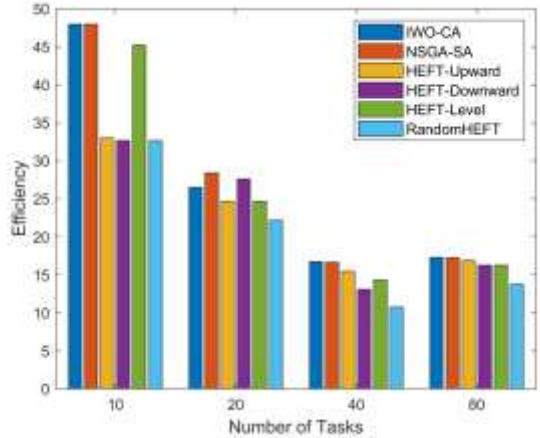

Fig.15: Efficiency(CCR=1)

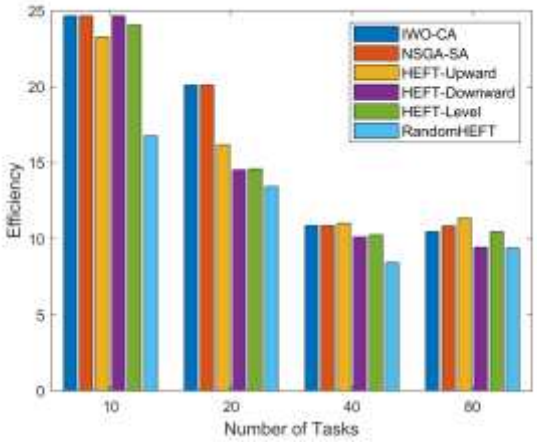

Fig.16: Efficiency(CCR=2)

The results shown in figures 15 and 16 show demonstrate that the suggested method significantly improves efficiency compared to existing methods. Table 3 shows, when the CCR = 1, that the proposed approach improves efficiency by 22.49% and 39.08%, respectively, compared to the HEFT-U and RandomHEFT methods. When the CCR value is 2, the rate will be 7.48% and 38.41%, respectively.
Figures 17 and 18 show the SLR results. The lower the value, the better.

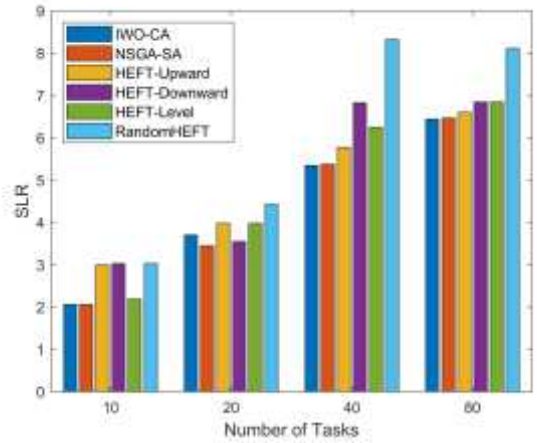

Fig.17: SLR(CCR=1)

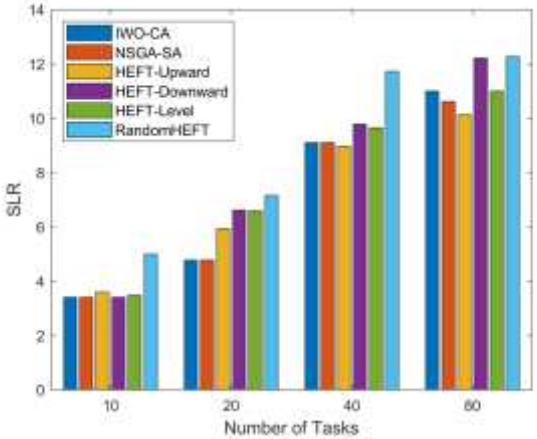

Fig.18: SLR(CCR=2)

In SLR, the denominator is the least computation of critical path tasks. There is no makespan less than the denominator of the SLR equation with any algorithm. As a result, the algorithm with the lowest SLR is the best. Figures 17 and 18 show that the suggested strategy outperforms the other methods tested using the SLR criterion. It is emphasized that as the number of tasks entered into the system increases, the amount of SLR gained from the suggested method will differ dramatically from that acquired from other algorithms. For the CCR = 1, table (3) shows that the suggested technique enhances the SLR parameter value by 14.3% and 27.36%, respectively, compared to the HEFT-D and RandomHEFT methods from right to left. For CCR = 2, this rate will be 12.83% and 22.79%, respectively.

7- Conclusions

One of the critical aims of in-network, cloud, and fog computing is to save energy and reduce response time. In this paper, we presented a method for job scheduling in an IoT-fog environment. The recommended solution employs hybrid algorithms NSGA-III and simulated annealing. The use of reference points in NSGA-III has led to the dispersion of population at the Pareto-front and the diversity of solutions. IETIF employs the Voltage Dynamics and Frequency Scale (DVFS) algorithm. Because it is one of the most important widely used methods for managing the heat generated by circuits and energy-saving consumption in microprocessors. The performance of IETIF was assessed in terms of energy consumption, response time, speedup, SLR, and efficiency. The suggested algorithm is compared to other methods such as HEFT-U, HEFT-D, HEFT-L, RandomHEFT, and IWO-CA. The simulation results show that our method outperforms the other algorithms. The essential factors for improvement in IETIF are the selection of the initial intelligent population and the employment of operators that, in addition to searching the entire problem space, lead to the generation of solutions that satisfy the problem conditions.

Author Contributions Statement

Mr. Nazari and Miss Soharabi implement the proposed method, Mr. Reza mohammadi, Mohammad Nassiri and moharram mansooorizadeh have written and edited the manuscript. All authors reviewed the manuscript.

Competing Interests

The authors have no competing interests to declare that are relevant to the content of this article.

Funding Declaration

The authors did not receive support from any organization for the submitted work.

References

[1] J. Chen *et al.*, "Big data challenge: a data management perspective," *Frontiers of computer Science,* vol. 7, no. 2, pp. 157-164, 2013.

[2] R. K. Naha *et al.*, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE access,* vol. 6, pp. 47980-48009, 2018.

[3] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach," *Journal of Network and Computer Applications,* p. 103333.2022 ,

[4] Y. Dai, Y. Lou, and X. Lu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-QoS constraints in cloud computing," in *2015 7th international conference on intelligent human-machine systems and cybernetics*, 2015, vol. 2: IEEE, pp. 428-431 .

[5] H. Wadhwa and R. Aron, "TRAM: Technique for resource allocation and management in fog computing environment," *The Journal of Supercomputing,* vol. 78, no. 1, pp. 667-690, 2022.

[6] F. Bonomi, R. Milito ,J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13-16 .

[7] M. S. U. Islam, A. Kumar, and Y.-C. Hu, "Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions," *Journal of Network and Computer Applications,* p. 103008, 2021.

[8] H. O. Hassan, S. Azizi, and M. Shojafar, "Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments," *IET communications,* vol. 14, no. 13, pp. 2117-2129, 2020.

[9] D. Senapati, A. Sarkar, and C. Karfa, "PRESTO: A Penalty-aware Real-time Scheduler for Task Graphs on Heterogeneous Platforms," *IEEE*

Transactions on Computers ,vol. 71, no. 2, pp. 421-435, 2021.

[10] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," *Concurrency and Computation: Practice and Experience,* vol. 32 ,no. 7, p. e5581, 2020.

[11] L. Mao, Y. Li, G. Peng, X. Xu, and W. Lin, "A multi-resource task scheduling algorithm for energy-performance trade-offs in green clouds," *Sustainable Computing: Informatics and Systems,* vol. 19, pp. 233-241, 2018.

[12] Y. Hao ,J. Cao, T. Ma, and S. Ji, "Adaptive energy-aware scheduling method in a meteorological cloud," *Future Generation Computer Systems,* vol. 101, pp. 1142-1157, 2019.

[13] S. Azizi, F. Khosroabadi, and M. Shojafar, "A priority-based service placement policy for Fog-Cloud computing systems," *Computational Methods for Differential Equations,* vol. 7, no. 4 (Special Issue), pp. 521-534, 2019.

[14] S. Omer, S. Azizi, M. Shojafar, and R. Tafazolli, "A priority, power and traffic-aware virtual machine placement of IoT applications in cloud data centers," *Journal of Systems Architecture,* vol. 115, p. 101996, 2021.

[15] S. Javanmardi, M. Shojafar, V. Persico, and A. Pescapè, "FPFTS: A joint fuzzy particle swarm optimization mobility-aware approach to fog task scheduling algorithm for Internet of Things devices," *Software: Practice and Experience,* 2020.

[16] 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.

[17] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A method based on the combination of laxity and ant colony system for cloud-fog task scheduling," *IEEE Access,* vol. 7, pp. 116218-116226, 2019.

[18] 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," *Applied Sciences,* vol. 9, no. 9, p. 1730, 2019.

[19] R. Xu *et al.*, "Improved particle swarm optimization based workflow scheduling in cloud-fog environment," in *International Conference on Business Process Management*, 2018: Springer, pp. 337-347 .

[20] T. Meyarivan, K. Deb, A. Pratap, and S. Agarwal, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans Evol Comput* , vol. 6, no. 2, pp. 182-197, 2002.

[21] I. O. Essiet, "Improved Evolutionary Algorithms with Application to Smart Grid Demand Response Management," University of Johannesburg, 2019 .

[22] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing,* vol. 8, no. 2, pp. 175-186, 2014.

[23] P. Hosseinioun, M. Kheirabadi, S. R. K. Tabbakh, and R. Ghaemi, "A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm," *Journal of Parallel and Distributed Computing,* vol. 143, pp. 88-96, 2020.

[24] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems,* vol. 13, no. 3, pp. 260-274, 2002.

[25] S. AlEbrahim and I. Ahmad, "Task scheduling for heterogeneous computing systems," *The Journal of Supercomputing,* vol. 73, no. 6, pp. 2313-2338, 2017.