A Hyper Heuristic Algorithm for Scheduling of Fog Networks

Sabihe Kabirzadeh, Dadmehr Rahbari, Mohsen Nickray
Department of Computer Engineering and Information Technology, university of Qom
Qom, Iran
s.kabirzadeh, d.rahbari@stu.qom.ac.ir, m.nickray@qom.ac.ir

Abstract—**Fog computing is a new computing structure that brings the cloud to the edge of the network. This structure is designed for applications that require a low latency. Fog computing has been proposed to improve cloud computing disadvantages. The system is faced with a variety of dynamic resources distributed and heterogeneous. Hence, scheduling and allocating resources is essential to maximize the use of these resources and the satisfaction of users. Classical algorithms are suitable for small scheduling problems, but the problem emerges in big scheduling problems. To improve the performance of the scheduling problem, heuristic algorithms are used. In this paper, we used the test and select technique to introduce a hyper-heuristic algorithm. We compare the proposed algorithm with several heuristic algorithms. The results show that our proposed algorithm improved the average energy consumption of 69.99% and cost 59.62% relative to the PSO, ACO, SA algorithms.**

I. INTRODUCTION

The Internet of Things (IoT) is a new concept in technology and communications scopes. In short, the IoT is a modern technology in which intelligent objects around us interact with each other to achieve common goals [1], and includes every online object like smart cameras, wearable sensors, environmental sensors, smart home appliances, vehicles, and etc. [2],[3].

The number of connected devices is now more than the number of people on earth. This number has reached about 9 billion and is expected to reach 50 Billion of devices in 2020. The IoT increases the quality to human life, but the use of the IoT produces massive amounts of data, which creates an excessive burden for data storage systems and analysis [4][5]. As a remedy, cloud computing is used to manage and control this massive amount of data produced by objects. Many applications, such as health monitoring application or intelligent traffic control application or games may need to get feedback in a short amount of time, and the latency caused by sending data to the cloud and then returning the response from the cloud to the operator of these programs has an adverse effect. Furthermore, the massive amount of data generated by some of these applications may impose heavy burdens on the network, and sending this volume of data to the cloud and then returning it is not desirable [6][7]. Hence, Bonomi presented a new concept called the fog computing in 2012 [8]. Fog computing extends clouds to the edge of the network and a solution to overcome the limitations.

Fog computing a distributed computing and is located between objects and the cloud. Fog computing architecture includes three layers: the bottom layer the layer nodes of IoT that includes smart devices, sensors and so on. The middle layer is the fog computing layer and includes tools such as gateways, routers, and switches. The last and highest layer, the cloud layer includes servers and data centers. Fig. 1 shows the architecture of fog network. Fog computing not an alternative

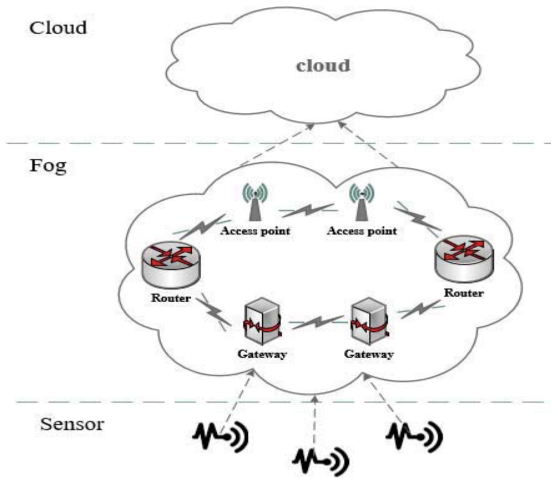


Fig. 1. Fog Computing Architecture

to cloud computing but it fixes cloud computing problems and increases its efficiency. Cloud computing and fog computing have common characteristics. However, fog computing has more features than cloud computing, including geographical distribution, real-time interaction, support mobility, heterogeneity, and interoperability [5]. When computing power is needed, multiple fog nodes can be implemented instead of a single fog node for computation, which increases scalability and flexibility. Fog computing has several advantages, including decrease latency time, decrease network traffic and energy efficiency, but due to the novelty, this concept also has challenges [9]. One of these challenges is associated with the allocation of resources and scheduling [10]. In fog computing, job scheduling problem means assigning a set of tasks to fog nodes located at the edge of the network [11]. Applications in the fog computing environment between sensor and cloud run on fog devices such as a gateway, switches, and so on. These resources are pervasive and variable. Therefore, an efficient scheduling and allocation of resources are essential to maximize the use of these resources and increase profit providers of fog and fog. In this paper, we are in search for hyper-heuristic algorithms for scheduling and allocating resources between fog nodes with the goal of maximizing the use of network resource's users [6]. Our main contributions to this paper are following:

1) We propose a scheduling method for application modules by hyper heuristic (HH) algorithm includes Genetic Algorithm(GA), Particle Swarm Optimization Algorithm(PSO), Ant Colony Optimization Algorithm(ACO) and Simulated Annealing Algorithm(SA).
2) We reduce the energy consumption and execution time in fog computing environment.

In this paper, we focus on the job scheduling problem in the fog computing environment. The rest of the paper is organized as follows: Section II reviews related work. The proposed job scheduling algorithm in Section III. In Section IV the results for a simulation and in Section V conclusions.

II. Related work

A scheduling method in distributed environments generally divided into three categories: Resource scheduling, workflow scheduling, and task scheduling [12]. Scheduling is defined as follows: finding an optimal solution for scheduling a set of task's $T = \{T_1, T_2, ..., T_n\}$ or workflow on a set of machines $M = \{M_1, M_2, ..., M_m\}$. The scheduling can be done by the deployment of a set of predefined constraints and objective functions [13]. The task is a small part of a work that must be performed within a specified time. One of the goals of task scheduling is to maximize the use of existing resources and minimize the waiting time on a job [14]. The aim of scheduling is to benefit two categories: service providers and service users. Service user interest corresponds to makespan, budget, deadline, security, and cost. On the other hand, service provider's objective is load balancing, resource utilization, and energy efficiency. The use of different objectives scheduling in various articles includes: makespan (40%), cost (14%), deadlines (14%), load balancing (14%) and budget (6%) [15]. Makespan is the end time of the last job. The aim is to minimize the time to complete the last task. In recent years, the problem of scheduling tasks is widely used in distributed computing systems.

Fog computing is a new concept, and there are only a few studies on scheduling in fog computing. Deng et al. [16] provide a scheduling methodology for the cloud-fog environment. This schedule is based on energy consumption and transmission delays. Intharawijitr et al. [17] have proposed a scheduling scheme aiming to reduce the amount of computation and latency in fog computing. Their scheduling is done based on three policies. In the first policy, the fog node is randomly assigned to execute a task. The second policy is based on the selection of the fog node, which generates the least latency according to the current state within the system, and the third policy of the fog node, which has the most remaining resources among other fog nodes, is selected to run the task.

In authors study, Bitam et al. [11] proposed a bio-inspired solution based on the algorithm of the Bees Life to solve the scheduling problem for the fog computing environment. This solution is based on the distribution over a set of tasks between all fog nodes.

Scheduling algorithms in systems such as clouds or the edges are divided into two categories. Traditional or classical algorithms (based on law) and intelligent algorithm [13],[18]. Classical algorithms are suitable for small scheduling problems, but the problem emerges in big scheduling tasks. Researchers have tried to obtain better solutions for large complex problems by efficient algorithms such as heuristic and meta-heuristic algorithms [19],[20].

A. Traditional algorithms

FCFS scheduling, when a new job arrives, the end of the queue placed. The first job from the beginning of the queue always runs first. Round-robin method is based on the FCFS method for scheduling tasks. Resources are assigned to tasks for fixed periods. The advantage to this approach is load balancing [12][21].

The Min-Min method, the smallest job of tasks available to select and assign it to the machine takes the least possible time to finish this job. This method increases completion time of all tasks so makespan increases. The method Max-Min, select the longest work among existing tasks and assign it to the fastest machines for the run [14][22].

B. Heuristic algorithms

The Security-aware and Budget-aware (SABA) algorithm [23] for scheduling in a multi-cloud defined. The algorithm contains three main steps. First step clustering and the next steps are to prioritize tasks and the final step of assigning data to the specific data center is based on the workflows constant data sets.

Multi-objective Heterogeneous Earliest Finish Time (MO-HEFT) [24] scheduling algorithm is known as a model of HEFT. A heuristic method based on a set of Pareto-based solutions. Enhanced IC-PCP with Replication (EIPR) algorithm [25], a scheduling and provisioning solution that uses the idle time of provisioned VMs and a budget surplus are paid to repeat the tasks to meet the deadline.

C. Meta-heuristic algorithms

PSO is a population-based random optimization algorithm. The dimension of the particles is equal to the number of tasks, and the position of the particle shows the mapping between the virtual machine and tasks. Some of the scheduling schemes using only the basic PSO algorithm [26][27], but others used improved [28][29][30]. Some of the scheme's scheduling workflow using the original algorithm GA. While the rest change it for the best results. Many of them produce best initial population to achieve better results [31],[32]. Szabo et al. [33] have proposed two chromosomes. A chromosome that is responsible for assigning nodes and a chromosome is responsible for the ordering.

Ant Colony Algorithm (ACO) by Marco Dorigo inspired from the behavior of some species of ants was introduced. Initially known as the ant system. Ants leave a material in the ground called pheromones to guide other ants. This behavior is used for solving optimization problems. ACO algorithm is used in many optimization problems, including scheduling. Liu et al. [34], have used the ACO method to deposit pheromones between virtual machines in order to achieve past utility of placing pheromone in physical machines. The algorithm is simulated in a homogeneous environment, and only CPU and memory resources are considered.

D. Hybrid heuristic algorithms

Guddeti et al. [35] a combination new bio-inspired algorithms for job scheduling and resource management are provided within the cloud computing environment. Here, using improved PSO tasks and its combination with biology-inspired algorithms, tasks are assigned to virtual machines. The proposed algorithm is a combination of modified PSO and CSO. Delavar et al. [36] offers a mix of GA, Best Fit, and Round Robin algorithms. Which aims to reduce the number of iterations that operators of the genetic algorithm have created. An optimal initial population is obtained by integrating of Best Fit and Round Robin algorithms.

E. Hyper heuristic algorithms

Tsai et al. [13] a heuristic scheduling algorithm, called the hyper-heuristic scheduling algorithm (HHSA), is provided to Find better scheduling solutions of the cloud computing environment. The low-level heuristic is used to find better candidates as solutions.

Gomez et al. [37], provides a multi-objective framework for making selections hyper-heuristics to solve the problem of two-dimensional bin packing. The solution includes a multi-objective evolutionary learning process, using genetic operators to generate a set of rules to represent hyper-heuristic.

III. Proposed algorithm

The nodes in the sensor networks are receiving data from their surroundings. The nodes send them to the gateways after receiving the data and then sends them to the fog devices. This data stored, processed in fog or sent to the cloud. A case study has several applications that are in fog. When the applications start in fog devices, a number of modules must be executed that needs to run the VM in the fog device, and the meta-heuristic algorithms are used to obtain the best allocation of VM to the modules. In the following, first are presented the algorithms that required to construct the hyper-heuristic method, which includes algorithms GA, PSO, ACO, and SA, and then explained the proposed algorithm based on hyper-heuristic algorithms. Each of the algorithms is used by a fitness function to evaluate the proper allocation of resources to the modules. Fitness function is defined for our proposed scheduling algorithms according to Formula (1).

$$Fitness = \frac{1}{TUC + BW} \tag{1}$$

Where: TUC defines the total CPU utilization and BW is bandwidth. By increasing these two values, the amount of fitness function decreases. Here we are looking to increase the value of the fitness function, so we need to minimize the amount of CPU utilization and bandwidth.

Genetic Algorithm: The Genetic Algorithm was defined in 1975 by Holland [38]. one of the best population-based algorithms in terms of performance and ease of use for different problems. The GA includes initialization of populations by VM list as $VMs = \{vm_1, vm_2, ..., vm_n\}$ for modules, se-lected operators, reproduction, crossover, and mutation. Chro-mosome also represents a solution that is generated randomly initialized. Based on the fitness function chromosome is se-lected and the single point crossover and mutation operations on it to generate new population done. This process is re-peated until sufficient children are produced. Unlike heuristic algorithms that use the objective function to choose the best solution, GA algorithm implements the fitness function for selecting the best solution.

Particle Swarm Optimization Algorithm: The particle swarm optimization algorithm (PSO) was introduced by Kennedy and Eberhart in 1995 [39]. Each particle as VM list is known by position, speed, and how to move into a search space. The evaluation of each particle is determined by the fitness function. At each repetition, the speed and position of each particle adapt itself to the best particle. PSO has been of great interest because of its simplicity and the low computational cost.

Ant Colony Optimization Algorithm: The ant colony optimization algorithm was inspired by the behavior of real ants in finding the shortest route between their colony and a source of food [40]. Ants that walk between the colony and the food source leave a material called pheromone in place and by increasing the movement of ants in one direction the pheromone intensity increases. ACO is useful for problem-solving discrete optimization methods that need to find a way to target. Each ant has a solution for all tasks assigned to the VMs. At first, the pheromone is a positive constant value. However, after each iteration ants update the pheromone amount. The solution is based on a certain fitness function. **Simulated Annealing Algorithm:** The simulated annealing algorithm [41] has three steps: 1) heating step. Its aim is to the increased thermal motion of the particles and its devi-ation from the equilibrium position. 2) The isothermal step which is based on heat exchange in the environment and avoids the constant temperature for the system. The system automatically goes to energy reduction, and when energy is minimized, the system reaches equilibrium. 3) Cooling step. The aim is that the thermal motion among the particles as $VMs = \{vm_1, vm_2, ..., vm_n\}$ are slipped and become more regular. The idea of the simulated annealing algorithm is that the heated solid is melted and then allowed to cool slowly to form a regular crystal solid. This tremendous flexible algorithm builds a local search and can successfully be applied to many real-life problems.

A. Hyper-heuristic scheduling

A hyper-heuristic is a heuristic search method that combines machine learning techniques, chooses, combines, generates, or matches many simple heuristics to solve computational search problems. To solve a problem, there may be multiple heuristics, each with disadvantages and advantages [42]. The main idea of hyper-heuristic is to combine the benefits and compensate for the weaknesses of simple heuristic. A hyper-heuristic (HH) method is a heuristic one used for selecting and producing hard computational problems (NP-hard). Since all hyper-heuristic algorithms aimed at finding a discovery heuristic among the heuristic of their own candidate can be used rather than repeating the process of finding a heuristic algorithm for an incoming workflow, a new algorithm that uses classification and clustering strategies on previous data to find the appropriate algorithm for workflow. Our proposed algorithm is done using the test and select techniques, and the best algorithm is selected among the candidate algorithms for the new workflow. Our proposed method consists of two phases: training phase and test phase. The pseudo-code of this algorithm is shown in Algorithm 1. Training phase: Initially, 64 different workflows enter the system. Then in Lines 4 through 7 of our choice algorithm includes GA [38], PSO [39] and ACO [40], SA [41] is implemented to allocate resources to modules in all workflow.

At line 8 of application, the intelligent monitoring system comes along with its modules. The intelligent monitoring system is described in Section III-C. At the lines of 9 to 11, the amount of consumed energy, the use of network and the cost of running each algorithm for each workflow are achieved after the completion of each algorithm. Then, the results are stored in a dataset, and for each workflow, the best algorithm is selected as a label.

Energy Consumption means the total energy consumed by the system. This energy is used by any of the network components such as fog, sensors, gateway, and so on. This energy consumption can be calculated by Formula (2). The amount of network utilization, cost and execution time of running any algorithm is also obtained from Formula (3), (4) and (5).

$$energy = CEC + (CT - LUUT) * HLU \tag{2}$$

CEC is current energy consumption. CT refers to current time. Also, LUUT returns the value of last utilization update time, and HLU refers to last utilization of the host. The energy consumed from the beginning of the simulation is zero. After running the simulation to obtain the energy consumed, the simulation time, which is the difference between the current system time and the last utilization updates time, is multiplied in the last utilization of host and eventually added to the amount of current energy consumed. This value is based on megajoule.

$$Networkusage = \frac{(TL * TS)}{MST} \tag{3}$$

In Formula (3), the values of TL and TS represent the total latency and total size of the tuple, respectively. Maximum Simulation Time Shown with MST.

$$Cost = CC + (CT - LUUT) * RPM * LU * TM \tag{4}$$

Where CC is current cost, CT denotes the current time, LUUT is the last utilization update time, RPM represents the rate per MIPS, LU is last utilization, and TM is total MIPS of the host. The cost of allocating resources includes costs (allocation of memory, bandwidth, storage space, and processor). At the beginning of the simulation, all costs are set and the initial cost is zero. After running the simulation of the updated values and the total simulation cost is obtained according to the Formula (4). The current time value of the system decreases from the value of the utilization update time, then this amount is multiplied by the rate of getting per million instructions per second and the last utilization system and the total of the million instructions per second per host, and the result adds to the current cost of the simulator. This value is based on the non-negative number.

$$ExecutionTime = CT - SST \tag{5}$$

In Formula (5), the values of CT represent the current time, and SST denotes the simulation start time. After completion of the simulation, the current time on the system will be reduced from the start time of the simulation until is achieved the simulation time. This value is in milliseconds and is calculated based on the simulator executive clock.

Test phase: line 15 represents new workflow enter the system. Line 16 E uclidean distance new sample obtained with examples inside the dataset. In line 17, the best algorithm is chosen. Then, energy consumption, network usage and total cost of implementing the algorithm on new workflow are calculated. Hyper-heuristic the algorithm calculates the total cost as the cost of the selected algorithm. Finally, results are returned. Our n ew method uses classified techniques of the incoming data set that contains examples of workflows that are labeled with the appropriate algorithm.

Algorithm 1 Hyper Heuristic Algorithm

```
1: Start
2: Starting Training Phase
3: Initialization nember of Area, number of Camera, Scheduling
4: Log in n=64 workflow
5: For $i = 1$ to 16
6: Run Candidate Algorithms in list(GA, PSO, ACO, SA)
7: EndFor
8: Calling the Application DCNSFog
9: For $i = 1$ to 64
10: Calculate energy consumption, network usage, execution time and total cost
11: EndFor
12: Save the results of step 10 as a dataset
13: End of Traning phase
14: Starting Testing Phase
15: Read new workflow
16: Calculate Euclidean distance new sample With dataset members
17: Choose the best algorithm based on the lowest Euclidean distance
18: Calling the Application DCNSFog
19: End of Testing Phase
20: Calculate energy consumption, network usage, execution time and total cost
21: Print the results
22: end
```

B. Test and select

A dataset has been generated during the heuristic algorithm training phase. Each row of this dataset has six columns, which include the number of areas, the number of cameras, energy consumption, a number of network resources used, the cost of execution, and the type of scheduling algorithm. In fact, these columns include network topology and outputs generated by scheduling algorithms. In the test phase, according to the network topology, a row of training datasets that have the least Euclidean distance as Formula (6) with input topology is selected and the last column runs as a scheduling algorithm.

$$Dist = \sqrt{(A_2 - A_1)^2 + (C_2 - C_1)^2} \tag{6}$$

A_1 and A_2 are the numbers of areas, also C_1 and C_2, the number of cameras in the training and testing phase.

C. Case Study:Intelligent surveillance

The smart surveillance system is designed to coordinate multiple cameras to monitor a specific area. Video surveillance/ object tracking software is a collection of distributed smart cameras that are capable of moving. As depicted in Fig. 2, the Intelligent Surveillance application consists of five major modules that include motion detector modules, an object detector, object tracker, interface and pan, tilting, and zoom control (PTZ). The camera sends the video stream to the motion detector module. This module sends it to the object detector by doing the filter on the video stream. This module identifies the object, identifies its position to the tracker module, and the appropriate PTZ is calculated and sent to the PTZ control. Finally, a fraction of the video streams containing each traced object is sent to the user's device [2]. The pseudo-code of DCNSFog application is shown in Algorithm 2.

Physical Topology: For the case study, we have considered a physical topology with four fog devices. Table I explained the configurations of the different types of fog devices used in the topology. The configuration of the sensors involved in the case study shows in the Table II. Here, the cameras that recording alive video feeds act as sensors and provides input data to the application.

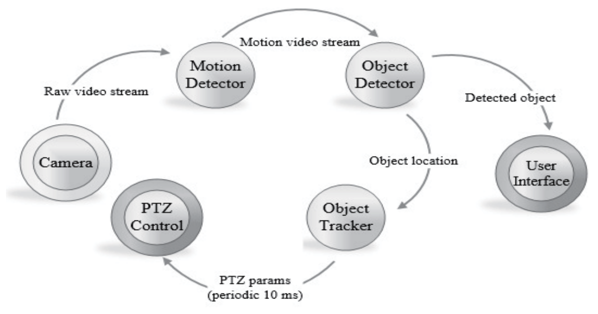


Fig. 2. Application Model of the Intelligent Surveillance Case Study

Algorithm 2 DCNSFog

```
1: Start.
2: Initialization numofarea, numofcamera, type scheduling.
3: Create new fogbroker and Create application.
4: Adding modules to the application model (object detector, motion detector, object
   tracker, user interface).
5: Connecting the application modules in the application model with edges.
6: Defining the input-output relationships of the application modules.
7: Defining application loops to monitor the latency
   of$Camera \rightarrow MotionDetector \rightarrow ObjectDetector \rightarrow$
   $ObjectTracker \& Userinterface \rightarrow PTZControl$
8: Add an application to fogbroker.
9: Creates the fog devices in the physical topology.
10: For $i = 0$ to $i \leq numofarea$
11: Add Area
12: For $i = 0$ to $i \leq numofcamera$
13: Add Camera
14: EndFor
15: EndFor
16: Initializing a module mapping.
17: For all fog device
18: IF fogdevice start with "m"
19: Fixing instance of the motion detector module to each smart camera.
20: EndIF
21: EndFor
22: Fixing instances of User Interface module in the cloud.
23: IF mode of deployment= cloud-based
24: placing all instances of object detector and object tracker module in the cloud.
25: EndIF
26: Create controller and Submit controller to application.
27: Start simulation: Cloudsim and Fog processing.
28: Stop simulation.
29: End.
```

TABLE I. Configuration of Fog devices for Intelligent Surveillance application

DEVICE TYPE	CPU GHz	RAM (GB)	POWER (W)
Cloud VM	3.0	4	107.339(M) 83.433(I)
WiFi Gateway	3.0	4	107.339(M) 83.433(I)
Smartphone	1.6	1	87.53(M) 82.44(I)
ISP Gateway	3.0	4	107.339(M) 83.433(I)

TABLE II. Configuration of sensor for Intelligent Surveillance

CPU Length	NW Length	Average Inter-arrival Time
1000 Million Instructions	20000 bytes	5 milliseconds

IV. Simulation and Result

In this section, we simulate fog computing for a case study. Then, we obtained energy consumption, network utilization, execution time and the total cost for this study.

A. Datasets and parameter settings

iFogSim [2] is a tool for simulating a fog computing environment. In this study, we used the iFogSim tool to simulate the workflow scheduling problem. The empirical analysis was conducted on a PC with Intel Core i5 CPU and 3GB of memory running Windows 10- 32bit. The initial population size is 64 for the training phase and the test phase 16. The number of checked areas has changed from one to four. Other parameters of the scheduling algorithms are shown in Table III. Each area has one to four smart cameras. These cameras monitor the area. These cameras connect to an area gateway, which is responsible for accessing the internet. Based on the above configuration, the physical topology is designed. In this topology, the cloud is the highest point and cameras, and other fog devices are at the edge of the network. In Table IV, are initialized the simulation parameters.

TABLE III. Algorithm Parameters of Fog Computing Scheduling

Algorithm	Parameters
GA	Population size = 10, Mutation rate = 0.5 Crossover rate = 0.9, Elitism = 10
PSO	Swarm size = 10 Acceleration rate = 2
ACO	Ant count = 10, Pheromone updating rate = 0.1 choosing probability = 0.85, Influence weights = 0.95
SA	Mutation rate = 0.3, Cooling rate = 0.05 Starting temperature = 1
HH	Train samples = 64 Test samples = 16

TABLE IV. Simulation Parameters

Parameters	Value
Fog Device Time Zone	10.0
Fog Device Cost	3.0
Fog Device Cost Per Memory	0.05
Fog Device Cost Per Storage	0.001
Fog Device Cost Per BandWidth	0.0

B. Energy consumption

The energy consumption results obtained from the implementation of the algorithm on various configurations are shown in Table VI. With increasing areas under surveillance, the

TABLE V. DESCRIPTION OF NETWORK LINKS IN THE PHYSICAL TOPOLOGY FOR INTELLIGENT SURVEILLANCE

Source	Destination	Latency (ms)
Camera	Area Switch	2
Area GW	ISP Gateway	2
ISP Gateway	Cloud DC	100

TABLE VI. AVERAGE OF ENERGY CONSUMPTION OF SCHEDULING METHODS BASED ON THE NUMBER OF AREAS, CAMERAS AND FOG DEVICES

Area	Camera	Fog Device	Normalized Average of Energy Consumption				
			GA	PSO	ACO	SA	HH
1	1	4	0.02	0.79	0.56	0.58	0.00
1	2	5	0.06	0.80	0.91	0.80	0.83
1	3	6	0.06	0.80	0.77	0.77	0.78
1	4	7	0.09	0.81	0.87	0.83	0.79
2	1	6	0.07	0.81	0.85	0.80	0.07
2	2	8	0.09	0.83	0.90	0.78	0.10
2	3	10	0.12	0.84	0.88	0.82	0.12
2	4	12	0.15	0.85	0.91	0.86	0.14
3	1	8	0.07	0.77	0.79	0.76	0.07
3	2	11	0.10	0.83	0.83	0.84	0.12
3	3	14	0.16	0.86	0.91	0.86	0.16
3	4	17	0.20	0.93	0.93	0.91	0.21
4	1	10	0.10	0.82	0.90	0.82	0.08
4	2	14	0.15	0.89	0.89	0.83	0.13
4	3	18	0.20	0.91	0.96	0.92	0.19
4	4	22	0.25	0.96	1.00	0.94	0.25

number of cameras and fog device also increases, and each of these devices needs the energy to continue their work, so with this increase. The total energy consumption of the system also increases. Hence, you can see in Fig. 3, the energy consumption of the proposed algorithm in different modes is less than SA, ACO, and PSO algorithms and better performance than the three mentioned algorithms. The consumed energies by GA and HH are almost the same but that of the GA algorithm is slightly smaller.

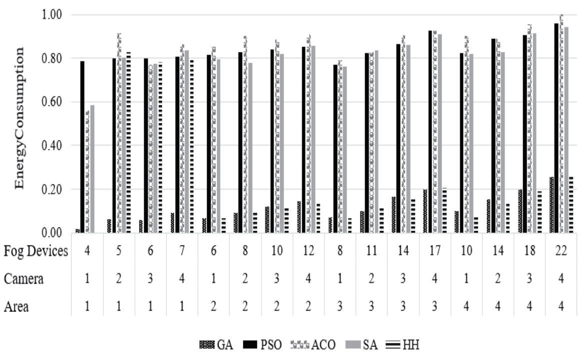


Fig. 3. Normalized Average of Energy Consumption for Different Methods

C. Network usage

The graph of network resource's usage is shown in Fig. 4. With increasing areas and the number of fog devices, increases the amount of use network resources. As you can see in the topologies, different algorithms are fully consumed network resources. The amount of network usage is calculated based on the Formula (3). In this way, for each topology, after executing the algorithm is obtained the sum of total latency and the total size of the tuples. And this amount is divided at the maximum simulation time. A number of network latencies between devices are listed on the Table V. This Table V is used to calculate the total latency.

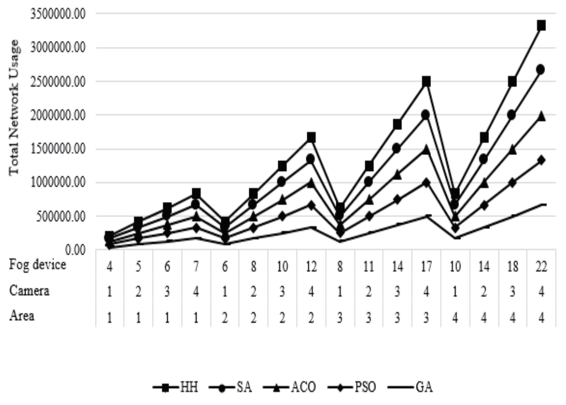
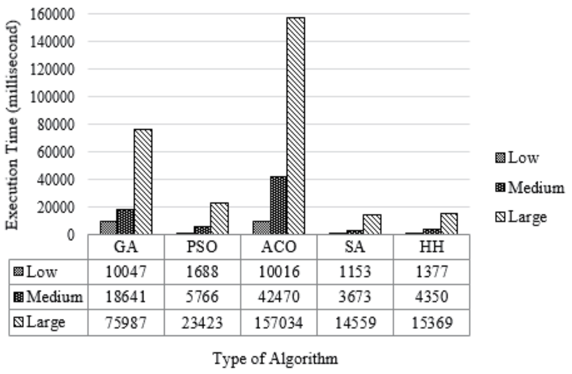


Fig. 4. Total Network Usage

D. Execution time

The simulation execution time is measured for the three different configurations based on Formula (5). The first configuration includes one area and a smart camera and is considered the lowest level. The average level has two areas and two smart cameras, and the highest level includes four areas and four smart cameras.

After executing different algorithms on these configurations, the execution time simulation is obtained for each of the algorithms. Fig. 5 shows the simulation execution time at these three levels of configuration. As you can see from the applied heuristic algorithms, the simulation execution time in the ACO algorithm has the highest value, and for the SA algorithms, it has the lowest value. Furthermore, the execution time of HH and SA is similar at all three topology levels. Therefore, our designed algorithm is the best algorithm in terms of execution time between the four algorithms.



	GA	PSO	ACO	SA	HH
▨ Low	10047	1688	10016	1153	1377
▩ Medium	18641	5766	42470	3673	4350
▧ Large	75987	23423	157034	14559	15369

Fig. 5. Execution Time of Scheduling Algorithms (millisecond)

E. Cost

The cost function is calculated according to Formula (4) in Section III for each algorithm. The results from the comparison of the total cost of the algorithms are shown in Fig. 6. The cost in fog computing includes operating costs. Suitable location and the optimal number of nodes play an important role in minimizing costs. The current cost is equal to the total simulation cost that this cost of the beginning of the simulation is equal to zero. After each run the simulator, the values are updated and can be calculated using Table IV. GA algorithm has the lowest cost, and the highest cost belongs to the ACO algorithm.

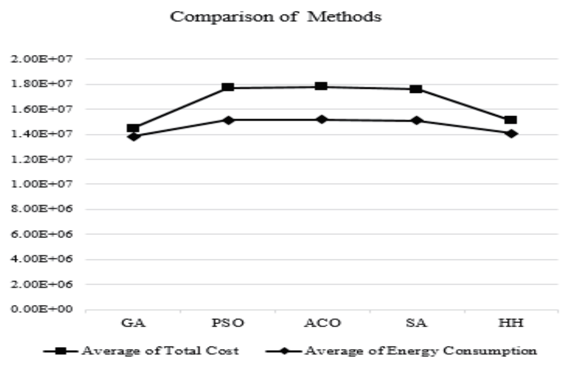


Fig. 6. Comparison of Scheduling Methods by Average of Energy Consumption and Total Cost and Train and Test Input Network Topology

V. Conclusions

This study presented a hyper-heuristic algorithm to find better solutions for the workflow scheduling problem in the fog computing environment. In this paper, a method based on a hyper-heuristic algorithm using a test and the select rule is presented. Our proposed algorithm in terms of average energy consumption 69.34% of the SA algorithm, 71.03% of the ACO algorithm and 69.60% of the PSO algorithm. Furthermore, average cost, compared with the SA algorithm is 58.84%, compared with the PSO algorithm 59.39% and the ACO algorithm is 60.65% improvement. This method reduced the simulation time and energy consumption by the size of a heuristic algorithm and increased the decision-making power for assigning resources with specific constraints to users, according to the type of workflows. Our proposed method at each step by choosing the best heuristic for the current workflow improves system performance. In this method, the algorithm is sensitive to the workflow that can be selected in accordance with the inputs of the appropriate heuristic algorithm. As a future work, we would like to explore and develop new methods for virtual machine's migration, resource provisioning and scheduling of applications in the internet of things based on fog networks.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey", *Computer networks*, vol. 54, no. 15, 2010, pp. 2787–2805.

[2] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments", 2016.

[3] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential", *Computer*, vol. 49, no. 8, 2016, pp. 112–116.

[4] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "Pre-fog: Iot trace based probabilistic resource estimation at fog.", IEEE, 2016, pp. 12–17.

[5] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live vm migration framework", *IEEE Access*, 2017.

[6] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed petri nets", *IEEE Internet of Things Journal*, 2017.

[7] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues", *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, 2016, pp. 2991–3005.

[8] F. Bonomi, R. Milito, J. Zhu, 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.

[9] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog Computing in Healthcare–A Review and Discussion", *IEEE Access*, 2017.

[10] C. Puliafito, E. Mingozzi, and G. Anastasi, "Fog computing for the internet of mobile things: issues and challenges", in *Smart Computing (SMARTCOMP),2017 IEEE International Conference on*, 2017, pp. 1–6.

[11] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm", *Enterprise Information Systems*, 2017, pp. 1–25.

[12] T. Mathew, K. C. Sekaran, and J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*. IEEE, 2014, pp. 658–664.

[13] C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A hyper-heuristic scheduling algorithm for cloud", *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, 2014, pp. 236–250.

[14] N. Patil and D. Aeloor, "A review-different scheduling algorithms in cloud computing environment", in *Intelligent Systems and Control (ISCO), 2017 11th International Conference on*, IEEE, 2017, pp. 182–185.

[15] P. Singh, M. Dutta, and N. Aggarwal, "A review of task scheduling based on meta-heuristics approach in cloud computing", *Knowledge and Information Systems*, 2017, pp. 1–51.

[16] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption", *IEEE Internet of Things Journal*, vol. 3, no. 6, 2016, pp. 1171–1181.

[17] K. Intharawijitr, K. Iida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5g cellular networks," in *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*, IEEE, 2016, pp. 1–4.

[18] N. K. Gondhi and A. Gupta, "Survey on machine learning based scheduling in cloud computing", in *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics and Swarm Intelligence*, ACM, 2017, pp. 57–61.

[19] P. Kaur and S. Mehta, "Resource provisioning and work flow scheduling in clouds using augmented shuffled frog leaping algorithm", *Journal of Parallel and Distributed Computing*, vol. 101, 2017, pp. 41–50.

[20] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments", *Concurrency and Computation: Practice and Experience*, vol. 29, no. 8, 2017.

[21] S. Nagadevi, K. Satyapriya, and D. Malathy, "A survey on economic cloud schedulers for optimized task scheduling", *International Journal of Advanced Engineering Technology*, vol. 4, no. 1, 2013, pp. 58–62.

[22] O. Elzeki, M. Reshad, and M. Elsoud, "Improved max-min algorithm in cloud computing", *International Journal of Computer Applications*, vol. 50, no. 12, 2012.

[23] L. Zeng, B. Veeravalli, and X. Li, "Saba: A security-aware and budget-aware workflow scheduling strategy in clouds," *Journal of parallel and Distributed computing*, vol. 75, 2015, pp. 141–151.

[24] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in amazon ec2", *Cluster computing*, vol. 17, no. 2, 2014, pp. 169–189.

[25] R. N. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, 2014, pp. 1787–1796.

[26] M. A. Rodriguez and R. Buyya, “Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds”, *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, 2014, pp. 222–235.

[27] K. Wu, “A tunable workflow scheduling algorithm based on particle swarm optimization for cloud computing”, 2014.

[28] A. Verma and S. Kaushal, “Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud”, in *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*, IEEE, 2014, pp. 1–6.

[29] C. Jianfang, C. Junjie, and Z. Qingshan, “An optimized scheduling algorithm on a cloud workflow using a discrete particle swarm”, *Cybernetics and Information Technologies*, vol. 14, no. 1, 2014, pp. 25–39.

[30] S. Chitra, B. Madhusudhanan, G. Sakthidharan, and P. Saravanan, “Local minima jump pso for workflow scheduling in cloud computing environments”, in *Advances in computer science and its applications*, Springer, 2014, pp. 1225–1234.

[31] T. Wang, Z. Liu, Y. Chen, Y. Xu, and X. Dai, “Load balancing task scheduling based on genetic algorithm in cloud computing”, in *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*, IEEE, 2014, pp. 146–152.

[32] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, “Ga-eti: An enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments”, *Journal of Computational Science*, 2016.

[33] C. Szabo, Q. Z. Sheng, T. Kroeger, Y. Zhang, and J. Yu, “Science in the cloud: allocation and execution of data-intensive scientific workflows”, *Journal of Grid Computing*, vol. 12, no. 2, 2014, pp. 245–264.

[34] X.-F. Liu, Z.-H. Zhan, K.-J. Du, and W.-N. Chen, “Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach”, in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ACM, 2014, pp. 41–48.

[35] R. M. Guddeti, R. Buyya *et al.*, “A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment”, *IEEE Transactions on Services Computing*, 2017.

[36] A. G. Delavar and Y. Aryan, “Hsga: a hybrid heuristic algorithm for workflow scheduling in cloud systems”, *Cluster computing*, vol. 17, no. 1, 2014, pp. 129–137.

[37] J. C. Gomez and H. Terashima-Marín, “Evolutionary hyper-heuristics for tackling bi-objective 2d bin packing problems”, *Genetic Programming and Evolvable Machines*, 2017, pp. 1–31.

[38] D. Whitley, “A genetic algorithm tutorial”, *Statistics and computing*, vol. 4, no. 2, 1994, pp. 65–85.

[39] J. Kennedy, “Particle swarm optimization”, in *Encyclopedia of machine learning*, Springer, 2011, pp. 760–766.

[40] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization”, *IEEE computational intelligence magazine*, vol. 1, no. 4, 2006, pp. 28–39.

[41] X. Liu and J. Liu, “A task scheduling based on simulated annealing algorithm in cloud computing”, *International Journal of Hybrid Information Technology*, vol. 9, no. 6, 2016, pp. 403–412.

[42] E. Ozcan, B. Bilgin, and E. Korkmaz, “Hill climbers and mutational heuristics in hyperheuristics”, *Parallel Problem Solving from Nature-PPSN IX*, 2006, pp.202–211.