Scheduling of Fog Networks with Optimized Knapsack by Symbiotic Organisms Search

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

***Abstract*—Internet of things as a concept uses wireless sensor networks that have limitations in power, storage, and delay when processing and sending data to the cloud. Fog computing as an extension of cloud services to the edge of the network reduces latency and traffic, so it is very useful in healthcare, wearables, intelligent transportation systems and smart cities. Scheduling is the NP-hard issues in fog computing. Edge devices due to proximity to sensors and clouds are capable of processing power and are beneficial for resource management algorithms. We present a knapsack-based scheduling optimized by symbiotic organisms search that is simulated in iFogsim as a standard simulator for fog computing. The results show improvements in the energy consumption by 18%, total network usage by 1.17%, execution cost by 15%, and sensor lifetime by 5% in our scheduling method are better than the FCFS (First Come First Served) and knapsack algorithms.**

I. Introduction

In today's world, many communication devices have built-in wireless sensors, with a large number of them distributed in a wide range and are still rising. Wireless sensor networks collect data on medical care systems, transportation, smart cities and more. These networks require real-time processing and decision-making as the infrastructure for the IoT (Internet of Things). In a cloud-based system, collected data should be transmitted to the cloud over a period of time and cost, and processed there. In fact, processing in the cloud has a lot of delays and bottlenecks for a large amount of data collected by end devices and sensors [1]. High delays in applications such as medical care, in very sensitive cases can cause a patient's death or cause a collision in the transportation system.

Fog computing places at the middle level between cloud and sensor nodes, so that data collection, processing, and storage can be done locally and only when the data is required to be transmitted to the cloud. The processing of the fog, due to the proximity of the sensors to the edge of the network, makes processing faster and also reduces network traffic. The fog computing with a large number of nodes has lower energy consumption than centralized cloud computing systems [2]. The fog computing architecture has a hierarchical arrangement of sensor nodes, edge devices, and cloud. Sensors are located in the lower geographic location at the bottom of the architecture and send the collected data to the up-level by gateway [3]. The actuators at the bottom of the fog architecture control the environment or change it. Operations performed in the fog include sensing and sending data, processing in the fog device, dividing an application into several modules, and allocating resources to them for execution. Applications are used to collect and store data in micro data centers as well as future analyzes and processes.

In iFogsim as an extension of Cloudsim, the application modules are defined as nodes and the connection between them as the edge in the fog network topology. Therefore, different applications of the fog network have different topologies. The client applications on the fog network can run as Cloudlet in virtual machines (VM) and provide flexibility, mobility, scalability, and elasticity [4]. Cloudlets in fog devices can do a variety of calculations, filter data to remove unnecessary items, and also assign a label to the data. So the tasks of the Cloudlets are executed in the VM. The VMs provide the resources to the applications by creating a multiple and shared status of the hardware. An optimal policy for allocating VMs will increase the productivity of resources inside the data center [5].

The scheduling process of programs in a distributed environment involves assigning resources to tasks in a specified order. Each VM is cost-effective, so configuring VMs for tasks is a challenge [6]. Performance measurements include processor efficiency, power consumption, runtime, and security. One of the solutions to this problem is the knapsack. Knapsack is an optimization problem in two ways, which includes the arbitrary and 0-1 [7]. Symbiotic organisms search (SOS) [8] is used to optimize the knapsack, we use it to reduce the time delay in optimal allocation of VMs in the fog network. Our scheduling strategy as KnapSOS is simulated in the fog network with two case studies. Our proposed simulation test is based on the collection of data related to the tracking of objects and humans, the transmission of data to the edge of the network, processing and sending them to the cloud. The simulation results are compared with the knapsack and FCFS algorithm. Our key contributions in this work are following:

1) Presentation of KnapSOS scheduling in fog computing.
2) Reduction of the delay and energy consumption.
3) Increase the total network usage.

In the following of the paper, Section II summarizes the related works with scheduling methods in cloud and fog networks. Fog network with two case studies as tracking applications is presented in Section III. In Section IV, we exactly explain the KnapSOS approach for scheduling. Details of the test and results are presented in Section V. The conclusion of this work is in Section VI.

II. Related work

Scheduling is the optimal use of CPU time and proper allocation of resources to programs. The main task of the scheduler is to decide which process to run in the next step by having a set of applicable processes. The scheduling objectives include cost, makespan, workload maximization, VM utiliza-

ISSN 2305-7254

tion, energy consumption, reliability awareness and security awareness. The Optimization strategies are heuristics, meta-heuristics [7], [9] and other methods as following. Resource models include VM leasing model, single or multiple VM type, single or multiple provider, intermediate data sharing model, data transfer and storage cost awareness, static, dynamic, subscription and time unit VM pricing model and VM provisioning delay [9], [10]. Scheduling strategies have different parameters and algorithms. We categorized these methods into traditional, heuristic, and meta-heuristic according to the following:

In scheduling of multilevel deadline-constrained scientific workflows, each provider offers a few heterogeneous VMs and a global storage service for data sharing [11]. One of the scheduling methods is earliest finish time, in which cost and makespan objectives are optimized. In [12],the authors analyzed commercial infrastructure as a cloud service. They used Pareto front as a tool of the decision support for the trade-off of appropriate solutions and proved that by increasing the makespan of 5% they could reduce the scheduling cost by half. In another way, task scheduling with fault-tolerant considered by bidding strategy for spot VMs and latest time to on-demand instances. The bidding starts at near spot price and increases during runtime, so that prices are close to demanding. The problem of this method is to reduce efficiency by using the cheapest spot VM [13]. In [14], the authors presented a real-time workflow scheduler based on hardware failures. Their algorithm is include shifting of the start time of the task without any change in its status and sub-set, resource scaling up for increasing capability of the VM processing or VM creation for adapting with the new task, and shrinking of processing capacity if the resource is idle for a certain period of time.

In [15], the authors studied the impact of quality of service on three scheduling method in fog network. These meth-ods include the concurrent, FCFS, and delay-priority. In the concurrent way, the arrival applications are allocated to a cloudlet regardless of usage capacity. In the FCFS method, the applications run in the order of entry, and if the processing power of the data center is less than the program request, then that program is placed in the scheduler queue. In delay-priority strategies, the applications scheduled based on lower delay. The researcher used iFogsim library with two games based on EEG signals as VSOT (video surveillance/object tracking application) and EEGTBG (EEG tractor beam game) and run those three expressed scheduling methods. The results show that greater delay and the number of modules per device of the concurrent method than FCFS and delay-priority. In the final comparison, the number of modules per device for delay priority method is approximately equal to all methods.

In the heuristic model, the solution of problem found by a number of rules. The classic heuristics include the first fit, best fit, and worst fit, so that the cloud and fog providers can run applications and tasks in large-scale and offloading states [9] by those methods. In [16], the tasks are scheduled with a heuristic algorithm, such that the objective function includes the makespan and the cost of executing the tasks. Results show increased efficiency and lower mandatory cost. In [17] a coalitional game based on cluster formation (CGBCF) proposed for radio access networks in the fog, which used cost by fetching contents missed in the cache, also it optimized the throughput of the network with a distributed user scheduling algorithm. Another method for vehicular cloud scheduling is a dynamic algorithm [18] by queue's length and response time parameters. Markov single server system scheduled tasks and modeled by stochastic Petri net. The simulation software of their work was OpenStack. In [19], parallel Real-time tasks in the heterogeneous network are scheduled by the heuristic method. The steps of algorithms include frequency selection, thread allocation, and Nonlinear programming also, objective parameters are based on the energy of the executive threads. In meta-heuristics model, general algorithms are designed to solve optimization problems [6]. Particle swarm optimization (PSO) is a used method for resource provisioning and scheduling that considers features such as the elastic provisioning and heterogeneity of unlimited compute resources as well as VM performance variation. In [20], bee's life algorithm (BLA) is proposed for the job scheduling problem in the fog network. Ant colony optimization (ACO) is a heuristic method for mobile cloud computing [21] that requires specific resources. This method executes offloaded Tasks in fog devices by delay, complete time, and energy consumption objectives.

In [22], the researchers have proposed the knapsack for task scheduling of parallel video transferring in the cloud, based on minimum complete time (MCT). They used the max min algorithm by estimating the maximum powerful computers and mapping to a number of segments and then scheduled segments by MCT algorithm. The results show that the max min algorithm is better than MCT in the execution time and the number of segments. The authors in [23], found the optimal orders of running tasks based on deadline and minimum cost by using of knapsack with dynamic programming. They considered small capacities of sub problems and obtained the best values of parameters include deadline in unpredictable tasks, the network congestion and the inaccurate task size. The different scheduling problem solved in [24] by optimized knapsack. They managed the smart grid using knapsack problem and consider the start timing objective. They used ACO to find the best solution, as they performed a mapping between multiple knapsacks and load scheduling. In [25], a resource scheduling in cloud solved by the combination of the knapsack and genetic algorithm (GA) with the fitness function include utilization of CPU, network throughput, and input/output rate of the disk. They decreased the energy consumption of physical machine and the number of migration than standard methods.

A variety of studies have been done on cloud in Cloudsim library. Some fog computing projects are simulated in Cloudsim [27] or different programming languages. iFogsim as an extension of Cloudsim is very suitable for fog computing simulation. Also, many of the meta-heuristic scheduling algorithms are very timely, so we use an optimized knapsack algorithm by SOS for fast execution and best results.

III. Applications and system models

Fog computing runs applications in fog devices between end devices and cloud. We use this paradigm with the benefits of cloud and edge for distributed data and low latency. IoT sensors are located in the lower layer of the architecture, which are responsible for receiving and transmitting data through the gateways to the higher layer, and also the actuators in the lowest level, are responsible for system controls. In fact, fog computing provides filtering and analysis of data by edge devices. Each application of fog network has a different

topology. The iFogsim simulator has a graphical user interface (GUI) module for designing custom and ready topologies [1]. Sensors, actuators, fog, cloud, and link elements can be added to the topology via this GUI. We create a case study with the new topology for healthcare environment and use a case study in iFogsim. These topologies can be read and executed by other modules in the simulator.

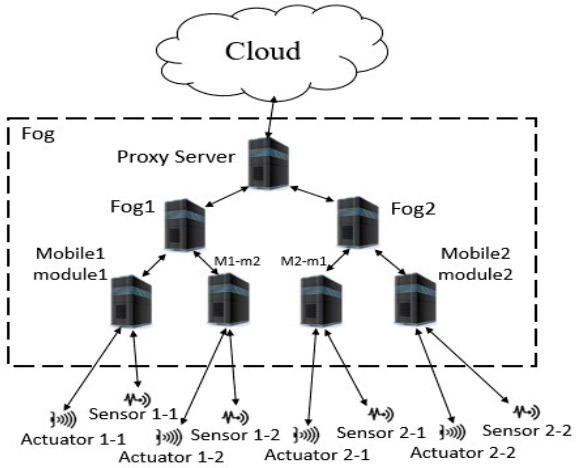


Fig. 1. Fog network topology

A. Case study 1: intelligent surveillance through distributed camera networks

This case is based on a distributed system of monitoring cameras in areas of healthcare, transportation, security, and manufacturing [1]. The application model for this case includes five functions.Object and motion detection in raw video streams by the camera. Object tracker for calculation of an optimal PTZ configuration. PTZ configuration is for adjusting the physical camera and serving as the actuator. User Interface, for sending a fraction of the tracked object to the user's device. The physical topology of case study III-A as DCNSGame is shown in Fig. 1. In this architecture, all cases in the dotted box are fog devices, M_i is module i and m_i is mobile i.

B. Case study 2: elderly human's activity detection

We present a novel application for elderly human's activity detection (EAHD) in the smart homes. The aging population have special health issues. The goals of such systems include monitoring and prediction of special health, daily activities, anomalous situations [26]. In order to identify the breakdown events, we simulate the gyroscope sensor for angular velocity and orientation detection and also accelerometer sensor for ambulatory activity. As EAHD process in the first step, data received from the sensors, in the second step, tracker unit recognizes the human activity or break down and in the third step, the result checks by controller unit and show the message in display monitor. The simulation of this application in fog network topology is based on Fig. 1.

C. Cloud concepts in fog

At the highest level of the fog computing architecture, there is a cloud. iFogsim uses the Cloudsim classes like Cloudlet, VM, data center, broker and so on [27] for communication with the cloud. The VM class, model a virtual machine interface by the host component. A host can simultaneously manage multiple virtual machines and allocate cores according to predetermined policies, in a time or spatial subscription. Each component of the virtual machine has access to the component that stores the characteristics of the virtual machine, such as memory, processor, storage space, and internal virtual machine timing policy that is derived from the VMScheduling abstract component. Cloudlet is for cloud modeling services generally featured in data centers have been implemented. Cloudsim shows the complexity of an application in terms of its computational requirements. Each program component has a preset command length. Each component of the program, in addition to its command length, also has the amount of data transmission that must be considered for the successful hosting of the program. The data center as a cloud hardware unit has a set of policies for allocating bandwidth, memory, storage devices to hosts, and VMs. The broker as an interface in the cloud carries out the task of mediating between users and service providers that require users based on the quality of their services.

IV. Proposed approach

The scheduling is NP-hard problem and those have high execution time. So, we propose a fast method for VM allocation to applications. The default resource scheduler in simulator equally divides a fog device's resources among all active application modules. In fact, by running of simulation and after a process as Algorithm 2, each application creates N modules as $\{M_1,\ M_2,\ M_3,\ \ldots,\ M_n\}$. Our proposed algorithm, schedule these modules for VM allocation. We change the original application scheduling policy by changing the updateAllocatedMips method in FogDevice class [1] with symbiotic organisms search based on knapsack algorithm and called it by KnapSOS.

A. Symbiotic Organisms Search

SOS algorithm is based on the two-paired relation of organisms, which can be found in the whole ecosystem by performing three steps. In the first phase, mutualism, two organisms together, both benefit. In the second phase, commensalism, one of the organisms benefits, and the another organism does not benefit or harm. In the third phase, parasitic, one of the organisms benefits, and another does not benefit [8]. We consider each VM as an organism in the ecosystem. According to Algorithm 1, there is three phase in SOS as the following.
Mutualism phase include 1: random selection of an organism, 2: calculate the mutual relationship vector (MV) and benefit factor (BF), $MV = \frac{(Xi+Xj)}{2}$, 3: (BF_1,BF_2) are random numbers either 1 or 2, 4: update organism X_i and X_j based on their mutual relationship, 5: $X_inew = X_i + rand(0,1) * (Xbest - MV * BF1)$, $X_jnew = X_j + rand(0,1) * (Xbest - MV * BF2)$, 6: calculate the fitness value of the updated organisms. If the updated organisms fitter than the previous then accept them.
Commensalism phase include 1: random selection of an organism X_j, 2: update organism X_i with the assist of organism X_j, 3: calculate fitness value of the new organism, 4: if the updated organism fitter than the previous the accept it.

Parasitism phase include 1: random selection of an organism X_j, 2: create a parasite (PV) from organism X_i, 3: calculate fitness value of the new organism, 4: if PV fitter than organism X_j then replaces organism X_j with PV.

Algorithm 1 KnapSOS scheduling

```
Initializing of VM scheduler.
Calculate Fitness by utilization of CPU and VM bandwidth.
Build the knapsack items
Start SOS:
Initialize ecosystem and maximum iteration
for i = 1 to MaximumIteration do
    for each organism in the ecosystem do
        Mutualism phase
        Commensalism phase
        Parasitism phase
        Update the best organism
    end for
end for
```

In EAHD Algorithm 2, the fog broker and application are created and for each camera and area create a fog device. The applications add to fog broker with the creation of fog devices. The modules of applications create in fog devices and related with together based on EAHD model. The next step is module mapping and the start of iFogsim with scheduling of modules for VM allocation by KnapSOS Algorithm 1.

Algorithm 2 EAHD with KnapSOS scheduling

1: Create fog broker and applications.
2: Add an application to fog broker.
3: **for** $i = 0$ to $i \leq numberofareas$ **do**
4: **for** $i = 0$ to $i \leq numberofcameras$ **do**
5: Create Fog device (Node name, MIPS, ram, upper bandwidth, lower bandwidth, busy power, idle power).
6: **end for**
7: **end for**
8: Add modules(sensor data stream, activity tracker, controller, display) to fog device.
9: Connecting the modules with edges.
10: Defining the input-output relationships and loops of the application modules. $SensorDataStream \rightarrow ActivityTracker \rightarrow Controller \rightarrow Display$
11: Initializing a module mapping.
12: Submit applications.
13: Start iFogsim.
14: Call **KnapSOS scheduling**
15: **for** Each VM **do**
16: **if** input application module is running in current VM **then** go to next VM.
17: **else** Allocate application module to VM.
18: **end if**
19: **end for**
20: Update energy consumption.
21: Stop iFogsim.
22: 18: Output result evaluation.

$$Fitness = \frac{1}{TUC + BW}. \quad (1)$$

The KnapSOS uses 1 as fitness function to allocate the VM to modules. In Formula (1), TUC is the total utilization of CPU for allocated VM to application module and BW is the bandwith of the application module. According to iFogsim, the brokers, applications, fog devices, and needed relations are created and then scheduler is called for resource allocation. In pseudo code 2, all steps are defined to execute our algorithm for each case study. The execution cost parameter is based on Formula (2) as the following:

$$Cost = TC + CC * LUUT * RPM * LU * TM. \quad (2)$$

In Formula (2), TC is the execution cost, CC is the CloudSim clock, LUUT is the last utilization update time, RPM is the rate per MIPS, LU is the last utilization, and TM is the total MIPS of the host. The total network usage is based on Formula (3).

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

In Formula (3), TL and TS are the total latency and the total size of tuple, and MST is the maximum simulation time. Also, the energy consumption of simulator is calculated for full topology of the network with Formula (4).

$$energy = CEC + (NT - LUUT) * HLU. \quad (4)$$

In Formula (4), CEC is the current energy consumption, NT is the now time, LUUT is the last utilization update time and HLU is the last utilization of host. The execution cost parameter is calculated in ifogsim by FogDevice class as Formula (4). We compare the proposed method with default FCFS algorithm in the simulator and knapsack algorithm using the above parameters. In order to place each component in the

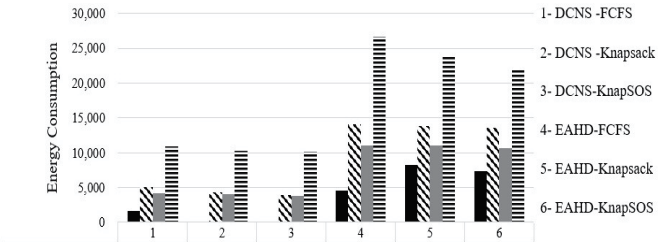
Algorithm 3 Create application

1: Add all **modules** to the application (module name, ram capacity).
2: Add all **edges** between modules for the application (source, destination module name, tuple CPU length, and direction).
3: Add **tuple** mapping (module name, input tuple type, output tuple).
4: Add Loops of **modules**.

Algorithm 4 Create fog device

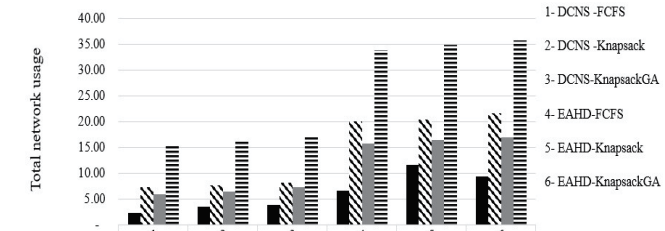
1: Create**processor** list.
2: Create **hosts** (Input, OS, VMs, cost, cost per storage).
3: Create **storage** list.
4: Set **latency**, upper and lower **bandwith**.
5: Mapping **application** to **modules**.

fog topology, a fog device is created as Algorithm 4. Tuples are a communication unit between components in iFogsim that are presented in a class. They are inherited from the Claudlet class in Cloudsim [1]. Each fog device is a micro data center with all the same features as the cloud data center with the lower processors and hosts. The fog devices are close to sensor nodes and cloud, so they can perform fast processing and local storing of the received data. The applications in fog devices have some modules are like cloudlet in the cloud.



	1	2	3	4	5	6
(Area=1, Mobile=1)	1,587	0	0	4,619	8,198	7,328
(Area=1, Mobile=2)	5,112	4,299	3,901	14,076	13,794	13,600
(Area=2, Mobile=1)	4,146	4,008	3,844	11,078	11,077	10,699
(Area=2, Mobile=2)	10,899	10,301	10,104	26,721	23,877	21,800

Fig. 2. Energy consumption of two case studies by FCFS, knapsack and KnapSOS scheduling method



	1	2	3	4	5	6
(Area=1, Mobile=1)	2.25	3.47	3.84	6.55	11.62	9.36
(Area=1, Mobile=2)	7.247	7.567	8.130	19.956	20.361	21.629
(Area=2, Mobile=1)	5.878	6.390	7.259	15.706	16.380	16.973
(Area=2, Mobile=2)	15.451	16.375	16.937	33.900	34.826	35.628

Fig. 3. Total network usage of two case studies by FCFS, knapsack and KnapSOS scheduling method

In Algorithm 2, to create the applications, modules or map applications to modules in directed graph, the FogDevice class is used. The input/output relationships are defined by tuples. In the final step, some loops are defined for monitoring latency of object trackers. Above steps show in Algorithm 3 and call in Algorithm 2. After changing of updateAllocatedMips function in FogDevice class, we run two case studies and analysis the iFogsim outputs. The KnapSOS algorithm help us to decrease the energy consumption and delay in fog computing.

V. Experiment and results

The simulation environment for this research is iFogsim library[1]. This Java language library has modules and classes needed to simulate fog computing. The iFogsim package and its classes are related to Cloudsim library. Our system for execution is PC with properties include Intel Core i5 CPU 2.67 Giga Hertz, 3 gigabyte RAM, and OS as Microsoft windows 10 32-bit. To run the novel scheduling algorithm, we use two case studies that presented in III-A,III-B and then compare the results with it. Also, We run the simulation by FCFS as an original scheduling method in the simulator, knapsack and KnapSOS algorithms, for each of the two case studies. There are 4 states of areas and mobile devices as $(Area, Mobile) = (1,1),(1,2),(2,1),(2,2)$. The initialization values of iFogsim entities as following:

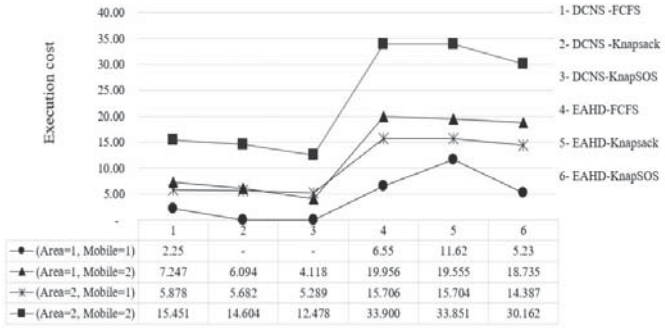
1: In **application modules**, MIPS is 1000, size in memory is 10 Megabyte, bandwidth is 1000 KBs, and ram capacity is 10.

2: In **fog devices**, storage capacity is 1 Gigabyte, bandwidth is 10000 KBS, the cost of using processing in this resource is 3.0, the cost of using memory in this resource is 0.05, the cost of using storage in this is 0.001.

3: In **proxy server**, the latency of connection between proxy server and cloud is 100 ms.

4: In **mobile devices**, latency of connection between mobiles and the parent fog device is 1 ms.

We run the simulation for 24 times as 4 states of $(area, mobile)$ and 3 methods include FCFS, Knapsack, and KnapSOS with 2 case study include DCNS and EAHD for each ot them. Our comparison is based on best results of the same configuration for case studies. In Fig. 2, we show the exact values of energy consumption in two case studies, so that all values are reduced by 13325117.71 value because it displays small values rather than large real values. The simulation results show that knapsack and KnapSOS methods obtain less energy consumption than the FCFS scheduling policy. As Fig.2



	1	2	3	4	5	6
(Area=1, Mobile=1)	2.25	-	-	6.55	11.62	5.23
(Area=1, Mobile=2)	7.247	6.094	4.118	19.956	19.555	18.735
(Area=2, Mobile=1)	5.878	5.682	5.289	15.706	15.704	14.387
(Area=2, Mobile=2)	15.451	14.604	12.478	33.900	33.851	30.162

Fig. 4. Comparison of execution cost in cloud of two case studies by FCFS, knapsack and KnapSOS scheduling method

DCNS case study III-A reduced the energy consumption by knapsack 14% and by KnapSOS 18%. This parameter for EAHD by knapsack has 1% increase and 5% reduction by KnapSOS. So the knapsack algorithm and KnapSOS are suitable and optimal than original FCFS allocation policy for VM allocation and scheduling in iFogsim simulator. In Fig. 3, we reduce all values from 2424.30 value as the average of total network usage in DCNS game has increased by knapsack 1.10% and by KnapSOS 1.17%. This parameter has increased for EAHD by knapsack 1.09% and by KnapSOS 1.10%. The total network usage parameter by knapsack and KnapSOS algorithm is more efficient than FCFS. As Fig. 4, the simulation cost in DCNS is decreased by knapsack 5% and KnapSOS 15%, also this parameter is decreased in EAHD by knapsack 1% and KnapSOS by 11%. In Fig. 4, we show that a few increase in execution cost but with the increase of area and activity tracker, the cost is reduced. For better illustration, all values in Fig. 4 are reduced by 7255.49 value. Comparing of two methods in this work with FCFS algorithm shows that our strategy has the lower energy consumption and execution cost in the cloud for two case studies in the fog computing and also our proposed algorithms increase the total network usage

TABLE I. Comparison of case studies.

Case study	Algorithm	Energy	Network Usage	Cost
DCNS	FCFS	Medium	Medium	Medium
	Knapsack	Low	Medium	Low
	KnapSOS	Low	Low	Low
EAHD	FCFS	High	High	High
	Knapsack	Medium	High	Medium
	KnapSOS	Low	Medium	Low

than original method of iFogsim. The evolutionary algorithms and meta-heuristic methods are good for objective optimization but those have higher execution time than knapsack algorithm, therefore, optimization of knapSOS algorithm is the better idea for scheduling of fog computing. The comparison of two case studies is shown in Table I. We show that the performance of FCFS, knapsack and KnapSOS algorithms by three levels as low, medium, and high values. For instance in DCNS case study III-A, the energy and execution cost are reduced than FCFS from medium to low and these parameters for EAHD case study III-B are decreased from high to medium and low.

VI. Conclusions

Because the clouds are located far away from the sensors and the data transmission has a lot of overhead and delay, so fog computing has an acceptable architecture for sensor applications. IoT applications as health care, vehicle, smart home, and so on have delay limitation. Fog computing with low latency and low traffic is the best framework for IoT environment. We optimize the iFogsim packages by the presentation of two new scheduling algorithms in fog devices. Our proposed methods are based on knapsack and KnapSOS algorithm that are simulated by two tracker case studies based on camera sensors with actuators. The results show that the average of parameters include energy consumption by knapsack 7% and KnapSOS 12%, total network usage by knapsack 1.09% and KnapSOS 1.14%, and the lifetime of sensors by 5% improve the original scheduling method in the simulator. Therefore, our proposed algorithm as KnapSOS for fog scheduling is better than the original FCFS algorithm and knapsack in iFogsim. In our future work, we are doing research the meta-heuristic algorithms for fog network scheduling with security considerations in other case studies.

References

[1] 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," *arXiv preprint arXiv:1606.02007*, 2016.

[2] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "Pre-fog: Iot trace based probabilistic resource estimation at fog," in *Consumer Communications and Networking Conference (CCNC), 2016 13th IEEE Annual*. IEEE, 2016, pp. 12–17.

[3] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach," *Future Generation Computer Systems*, 2017.

[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, 2009.

[5] P. Gupta and S. P. Ghrera, "Trust and deadline aware scheduling algorithm for cloud infrastructure using ant colony optimization," in *Innovation and Challenges in Cyber Security (ICICCS-INBUSH), 2016 International Conference on*. IEEE, 2016, pp. 187–191.

[6] 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, pp. 222–235, 2014.

[7] J. Lv, X. Wang, M. Huang, H. Cheng, and F. Li, "Solving 0-1 knapsack problem by greedy degree and expectation efficiency," *Applied Soft Computing*, vol. 41, pp. 94–103, 2016.

[8] M.-Y. Cheng and D. Prayogo, "Symbiotic organisms search: a new metaheuristic optimization algorithm," *Computers & Structures*, vol. 139, pp. 98–112, 2014.

[9] 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.

[10] M. E. Frincu, S. Genaud, and J. Gossa, "Comparing provisioning and scheduling strategies for workflows on clouds," in *Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW), 2013 IEEE 27th International*. IEEE, 2013, pp. 2101–2110.

[11] M. Malawski, K. Figiela, M. Bubak, E. Deelman, and J. Nabrzyski, "Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization," *Scientific Programming*, vol. 2015, p. 5, 2015.

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

[13] D. Poola, K. Ramamohanarao, and R. Buyya, "Fault-tolerant workflow scheduling using spot instances on clouds," *Procedia Computer Science*, vol. 29, pp. 523–533, 2014.

[14] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang, and L. Liu, "Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 12, pp. 3501–3517, 2016.

[15] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[16] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *Network Operations and Management Symposium (APNOMS), 2016 18th Asia-Pacific*. IEEE, 2016, pp. 1–4.

[17] Y. Sun, T. Dang, and J. Zhou, "User scheduling and cluster formation in fog computing based radio access networks," in *Ubiquitous Wireless Broadband (ICUWB), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–4.

[18] X. Chen and L. Wang, "Exploring fog computing-based adaptive vehicular data scheduling policies through a compositional formal method pepa," *IEEE Communications Letters*, vol. 21, no. 4, pp. 745–748, 2017.

[19] H.-E. Zahaf, A. E. H. Benyamina, R. Olejnik, and G. Lipari, "Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms," *Journal of Systems Architecture*, vol. 74, pp. 46–60, 2017.

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

[21] T. Wang, X. Wei, C. Tang, and J. Fan, "Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints," *Peer-to-Peer Networking and Applications*, pp. 1–15, 2017.

[22] F. Lao, X. Zhang, and Z. Guo, "Parallelizing video transcoding using map-reduce-based cloud computing," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 2905–2908.

[23] M. A. Rodriguez and R. Buyya, "A responsive knapsack-based algorithm for resource provisioning and scheduling of scientific workflows in clouds," in *Parallel Processing (ICPP), 2015 44th International Conference on*. IEEE, 2015, pp. 839–848.

[24] S. Rahim, S. A. Khan, N. Javaid, N. Shaheen, Z. Iqbal, and G. Rehman, "Towards multiple knapsack problem approach for home energy management in smart grid," in *Network-Based Information Systems (NBiS), 2015 18th International Conference on*. IEEE, 2015, pp. 48–52.

[25] S. Chen, J. Wu, and Z. Lu, "A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness," in *Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 177–184.

[26] Q. Ni, A. B. García Hernando, and I. P. de la Cruz, "The elderly's independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development," *Sensors*, vol. 15, no. 5, pp. 11 312–11 362, 2015.

[27] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.