

Bandwidth Allocation Algorithm for Makespan Optimization in a Fog-Cloud Environment: Monitoring Application

Bentabet Dougani, Abdeslem Dennai

Abstract

Fog computing technology has emerged to handle a large amount of data generated by the Internet of Things (IoT) terminals and cope with latency-sensitive application requests by allocating computation and storage resources at the edge of the Internet. In many IoT applications, the data acquisition procedures must apply the Directed Acyclic Graph (DAG) to get real-time results. The principal goal of DAG scheduling is to reduce total completion time without breaking priority constraints by properly allocating tasks to processors and arranging task execution sequencing. In this paper, we propose a bandwidth-aware workflow allocation (BW-AWA) that schedules tasks by priority to the resource and optimizes the total execution time (Makespan) in the entire computing system. The task allocation process needs to consider the dependency between tasks. The proposed approach is tested with a monitoring application case study, and the results are compared to well-known approaches to demonstrate its effectiveness in optimizing the Makespan.

Keywords: Cloud Computing, Fog Computing, IoT, List scheduling, DAG, Makespan.

MSC 2020: 68, 68M18, 68M20.

1 Introduction

With the Internet of Things emergence, real-time data access, processing, and storage require high bandwidth and low latency [1]. The

traditional Cloud Computing limitations in terms of working principle and low bandwidth may cause additional communication costs and latency. Therefore, Fog Computing technology has emerged and developed rapidly. A large amount of local data does not need to be uploaded to the Cloud for processing. The basic principle of Fog Computing is to process and analyze data on the data-generating side at the network Edge and provide services to the requests, thus effectively reducing the time delay generated by network transmission [2].

Fog Computing has some challenges. Computing nodes have limited resources, and tasks need to be distributed and scheduled according to the type and scale of actual tasks to avoid excessive load on some computing nodes, which affects system performance. The scheduling strategy aims to optimize the use of resources and the overall performance of task processing in the Fog Computing environment [3]. The performance objective can be achieved by using the allocated resources efficiently. Real-time monitoring of events in an environment is achievable through intelligent monitoring applications composed of multiple tasks cooperating in workflow models. Since the computerization of certain tasks in many IoT applications requires the result data of previous tasks, the workflow can be abstracted and modeled using a Directed Acyclic Graph (DAG), where the graph nodes represent sub-tasks, and the lines between nodes represent the priority constraint relationships between subtasks. The scheduling problem becomes even more complicated when computing systems are heterogeneous.

The execution time is the most important objective in the scheduling policy for the user in the real-time environment because the submitted workflow task has an urgent need for completion time. Since Fog nodes have limited processing capacity, the scheduling technique should consider the selection of the optimal processors while allocating the tasks. This paper presents workflow services for IoT applications in a Fog-Cloud Computing environment. We will propose a new list scheduling algorithm based on heuristics such as the most efficient and best-accepted category. This proposal algorithm will optimize the total execution time. Here are some factors considered when scheduling tasks:

- The algorithm is based on a task scheduling strategy for the work-

flow application in a heterogeneous Fog-Cloud computing environment.

- Calculate the priority of each task and according to the ranks, determine the order of the tasks.
- Improve the resource allocation phase: During the resource allocation phase, the algorithm will scan all the nodes for each task in order to find the node which has the most optimal bandwidth rate.

To simulate our scheduling algorithm, we applied it to dependent tasks case. This case contains the data collection related to IoT devices and their transfer to Fog nodes for processing and storage in the Cloud.

To evaluate the performance of our proposed algorithm by optimizing the Makespan, we have implemented it and run a set of experiments by modifying the number of tasks and the number of Fog nodes. The results obtained are compared with those given by the algorithms: FCFS (First Come First Serve), Popularity, Heterogeneous Earliest Finish Time (HEFT), and Critical Path On a Processor (CPOP).

The rest of this paper is structured as follows: Section 2 presents a synthesis of related works; Section 3 describes our proposed approach and addresses the task scheduling problem in a Fog-Cloud environment with our scheduling algorithm; the case study and the results of the experiments are presented in Section 4, before concluding and describing future work for the continuation of this research in Section 5.

2 Related work

In the literature, the task scheduling performance in heterogeneous computing systems has been evaluated in many types of research. A relevant solution for task scheduling methods uses optimization algorithms and performance metrics in real Fog scenarios to solve practical problems such as industry and factory [4], [5], transportation [6]–[8], and video processing [9]–[11]. Of course, the modeling and optimization algorithms' goals are different for different applications in different fields.

The Heuristic-based scheduling algorithms can be divided into three categories: List scheduling [12], clustering scheduling [13], and duplica-

tion scheduling [14]. The authors in [15] investigated the QoS (Quality of Service) parameters in a Fog network using three scheduling methods: Concurrent, FCFS, and delay-priority. The arriving tasks in the Concurrent method are assigned independently of consumption capacity in a concurrent manner. The tasks in the FCFS technique execute in the order of entry, and if the data center's processing power is less than the task request, the task is placed in the scheduler queue. Tasks are scheduled based on reduced latency in the delay-priority technique. The concurrent technique has a longer latency than FCFS and delay-priority methods, according to the results. However, these three strategies were not efficient for many problem instances. The workload can be divided into independent and dependent tasks based on the correlation between the scheduled tasks. Surely, for these different workloads, different algorithms are applicable [16].

Zeng et al. [17] studied the problem of minimizing the execution time of independent tasks by considering the placement of images in conjunction with task scheduling by assigning tasks and balancing the load among user devices and Fog devices, but without considering specific QoS requirements for the applications to be deployed. Therefore, developing an approximate algorithm is a good choice compared to the exact method.

Bitam et al. [18] compared a novel optimization algorithm named "Bees Life Algorithm" (BLA) with Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) algorithms in IoT-Fog environments. The results of this algorithm show a good performance of execution time and allocated memory.

In [19], a study of task scheduling for the IoT-based bag-of-tasks application in Fog and Cloud environment, Nguyen and colleagues proposed a GA algorithm to reduce the time required to complete the work. The authors have obtained better results than the BLA algorithm and MPSO (Modified PSO) algorithm, but the latency factor needs to be considered.

Rahbari and Nickray [20] proposed a knapsack-based scheduling algorithm that is optimized by symbiotic organisms and includes CPU and network usages in the fitness function. The proposed scenario was tested using iFogSim simulator. Compared to FCFS and Ordinary

Knapsack techniques, it offers gains in energy consumption and total network usage.

Jamil et al. [21] introduced a new Fog computing scheduler that may support IoE (Internet of Experience) service provisioning and therefore reduce delay and network use. A use case was also conducted with the goal of optimizing the task scheduling from end devices on Fog nodes and successfully addressing tasks on available resources on each Fog node. Latency and energy consumption were used as performance indicators in their study. They used iFogSim simulator to evaluate the suggested scheduling method in comparison to existing well-known approaches. Their results showed that the proposed latency and network utilization are more improved compared to the FCFS technique.

In [22], authors investigated an immune scheduling network-based task scheduling strategy in a Fog-Cloud environment. The suggested method uses the capability of distributed schedulers to produce efficient strategies for dealing with overloaded computing nodes and minimizing task completion times. According to the results, the proposed method outperforms the other compared strategies. Recently, exploiting iFogSim simulator, Guerrero et al. in [24], focus on resource provisioning in Fog environments. They propose a lightweight decentralized service placement policy for performance optimization in Fog computing to optimize service placement and reduce latency and network usage.

However, Intelligent algorithms are suitable for complex problems with strong constraints and multiple objectives. They are highly scalable, but they are difficult to be applied to scenarios with high real-time requirements, such as distributed online problems. While the number of workflow tasks increases, the scheduling time using intelligent algorithms will also increase. Since the computing system has limited resources, such as computation, storage, bandwidth, and battery power, it is more realistic to give the scheduling optimization model with limited resources as constraints.

Pham and Huh [23] focused on task scheduling in Fog-Cloud systems in order to find a balance between task executing times and cloud resource budgets. In their investigation, a DAG is applied to define the entering workflow. They show the effectiveness of collaboration be-

tween Fog nodes and rented Cloud nodes to run large-scale offloading workloads for the end user.

Topcuoglu et al. [12] proposed HEFT and CPOP as list scheduling algorithms under precedence constraint and dependency between tasks. In the first phase, HEFT calculates the priority of the task and ranks them in ascending order. A task ranking depends on the upward rank (rank u). This value is calculated for each task based on the execution and communication cost task. Then the resource with the lowest completion time is chosen to execute the task. But this is without taking into consideration the resource computation costs when executing the allocated task. CPOP is in the same category as the previous one. It tries to reduce the execution time by targeting the critical path. The algorithm creates a list of tasks in priority order. In accordance with HEFT, CPOP follows the same while considering another variable, the downward rank of nodes. The sum of the upward rank (rank u) and downward rank (rank d) for each node determines the tasks that are part of the critical path. Thus, we will have two categories: critical tasks, and the other tasks. The critical tasks will be processed with priority before the other tasks. However, the scheduling length may further rise as a result of the fact that all critical tasks are assigned to the resource with high processing capacity, thereby causing load imbalance among the computing nodes.

Table 1 resumes the main related works with a focus on the strategy, the scenario used, and the limitations. Workflow scheduling is an NP-complete problem resource [25]. Therefore, our paper proposes a workflow scheduling algorithm in a heterogeneous Fog-Cloud infrastructure based on a list scheduling algorithm. It sorts the tasks by priority. Then it selects the highest to the lowest priority task from the tasks set to be scheduled. The determination of priority is related to task ranking but the resource allocation deals with the bandwidth rate (bandrate) of each task in all processing nodes with the aim to select the optimal computing node which provides the optimal band rate. BW-AWA can increase task scheduling effectiveness when it is used to schedule the lower bandrate task in the high Bandwidth computing resource in the Fog server. This algorithm can therefore enable to optimize the makespan in the Fog-Cloud computing system.

Table 1. Related Work

Authors	Algorithms	Applica- tion	Limitations
Bitten- court et al [15]	FCFS, Delay- Priority	Surveillance Application	<ul style="list-style-type: none"> • Unclear Task Rank- ing • High Execution Time
Zeng et al [17]	Mixed-Integer Non Linear	General Case	<ul style="list-style-type: none"> • Homogeneous Nodes • High Complexity • No Task Ranking
Bitam et al [18]	BLA	Healthcare System	<ul style="list-style-type: none"> • High Complexity • No Task Ranking
Nguyen et al [19]	Based Genetic Algorithm	General Dataset	<ul style="list-style-type: none"> • High Complexity • No Task Ranking
Jamil et al [21]	SJF	Healthcare System	<ul style="list-style-type: none"> • High Execution Time • No Task Ranking
Wang et al [22]	Immune Scheduling	E immune System	<ul style="list-style-type: none"> • No Task Ranking • High Execution Time
Rahbari et al [20]	Knapsac	Surveillance Application	<ul style="list-style-type: none"> • No Task Ranking • High Execution Time
Topcuoglu et al [12]	HEFT-CPOP	Task Graph	<ul style="list-style-type: none"> • The issue of this pa- per does not target Fog-Cloud.
Guerrero et al [24]	Popularity	Sock Shop Demo	<ul style="list-style-type: none"> • Pre-fixed Task Ranking • High Execution Time
Pham et al [23]	HEFT-based Algorithm	General Case	<ul style="list-style-type: none"> • No Delay-Sensitive
Our Proposed	Bw-AWA Algorithm	Monitoring Application	<ul style="list-style-type: none"> • Optimizing Makespan • Task Ranking

3 Proposed Approach

3.1 System Model

The Fog-Cloud environment model is composed of heterogeneous computational resources. We define the Processor Graph (\mathcal{PG}) as a Workflow graph that $\mathcal{PG} = \{Q, L\}$, where: $Q = \langle q_1, q_2, \dots, q_m \rangle$ represents all the computational nodes, including Cloud server, the set of Fog nodes, and IoT devices. L defines the interconnection between computational nodes (i.e., the interconnection between q_i and q_j , noted: $q_i \leftrightarrow q_j \in L$).

We define computational nodes: $\vec{Q}(q_j) = \{Q_C \cup Q_F \cup Q_D\}$, respectively the set of Cloud server, the set of Fog nodes, and the set of end devices. Cloud technology may provide high processing capabilities, but it suffers from high latency and real-time processing limitations. Fog Computing is a distributed computing model that collects data at the edge of the network and processes the data in real time. The benefits of Fog computing include reducing bandwidth usage, optimizing data performance, and reducing latency.

Each node ($q_i \in Q$) has the attributes below:

The computational capacity of network node in *MIPS* (Million Instructions Per Second).

The memory (Ram) capacity in *Megabyte*.

The communication links between resources have two bandwidth levels: $UpBw(Kb/s)$, and $DownBw(Kb/s)$, respectively, represent Upper and Down bandwidth for each node.

3.2 Application Model

The IoT application consists of dependents tasks as Workflow Graph: $\mathcal{TG} = (\mathcal{MP}, \mathcal{EP}, \vec{Vmp}, \vec{VEp})$, where \mathcal{MP} denotes the set of n dependents task nodes M_1, M_2, \dots, M_n , each node M_i representing a task in the Workflow graph. \mathcal{EP} denotes the set of k directed edges between task nodes $E_j = E_1, E_1, \dots, E_K$. Specifically, each task $M_i \in \mathcal{MP}$ must be assigned to one resource $q_j \in Q$ with respecting the dependency constraints. The set $Succ(M_i)$ denotes the successors of task (M_i). The set $Pred(M_i)$ denotes the predecessors of task (M_i). If $Pred(M_i) = \emptyset$, then the task is called an entry task, and the task without a successor is called an exit task. Each task $M_i \in \mathcal{MP}$ has a computation weight,

and Each edge $E_j \in \mathcal{EP}$ has a communication weight representing the amount of data between two tasks (M_i, M_k) .

3.3 Proposed Solution

The basic idea is to generate a task scheduling list by assigning a priority to the task in the workflow, sorting it according to the priority, and allocating each ranked task to the optimal resource which is determined based on the bandwidth factor. Below is the diagram (see Figure.1) of our proposed solution:

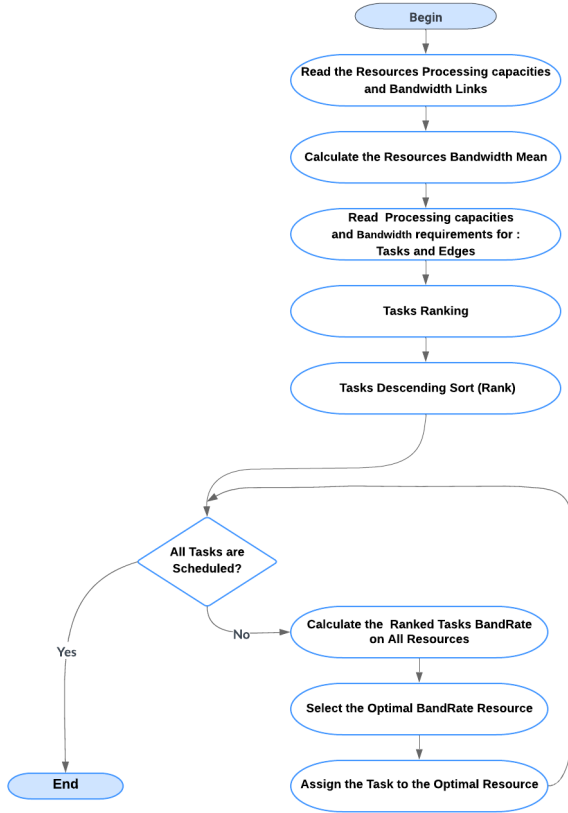


Figure 1. Our Proposed solution

Let $\overrightarrow{Vmp}(M_i)$ represent requirements of task node $M_i \in \mathcal{MP}$, and

$\overrightarrow{VEp}(E_j)$ is the characteristics for each edge in \mathcal{EP} .

For each task node $M_i \in \mathcal{MP}$, $\overrightarrow{Vmp}(M_i) = \langle Com_c(MIPS), Ram(Mb), \text{ and } mBW(Kb) \rangle$, respectively, representing the Computation cost, the memory required amount by the task, and the task node Bandwidth requirements. Tuples are the communication units between task nodes that represent the encapsulated $Data_{i,j}$ transferred between task nodes (i.e., between M_i and M_j). The IoT application receives the incoming tuples from the IoT Sensors which represent the Workflow graph source.

$$\overrightarrow{VEp}(E_j) = \langle T_{cc}(MIPS), BwE(Kb) \rangle,$$

where T_{cc} represent the Communication cost, and $BwE(M_i, M_j)$ is the bandwidth levels of E_j (M_i, M_j).

3.3.1 Task Ranking Phase

The Rank of each task is determined by three attributes computation: cost Com_c , encapsulated data amount in Tuple as communication cost T_{cc} (see Eq(1)), and predecessor task ranking $Rank(M_j)$ [26].

$$Rank(M_i) = \begin{cases} M_i, & \text{If } M_i = M_{exit} \\ Com_c(M_i) + \min_{M_j \in pred(M_i)} (T_{cc}(M_j) + Rank(M_j)), & \text{otherwise} \end{cases} \quad (1)$$

We define $Com_c(M_i)$ as the processing cost needed for the task M_i , and $T_{cc}(M_j)$ represents the data encapsulated amount in the Tuple (out) from the direct predecessor M_j to M_i , where the rank of the direct predecessor task is $Rank(M_j)$, and the rank value related to the exit task is $M_i = M_{exit}$.

3.3.2 Task Sorting Phase and Allocation Resources

In this phase, all priority of tasks is calculated and arranged in descending order from high to low. The task scheduling phase selects the task with respect to dependency according to the task rank list. The tasks will be unstacked one by one. The algorithm will scan the resources set for each task to find the resource that will optimize the bandwidth rate. The computing resources have a double link (full-duplex). Let

us note that the set of ($Mean_{Bw}$) represents the Average bandwidth of nodes in the Fog-Cloud architecture as shown in Eq(2):

$$Mean_{Bw}(q_k) = [upBw(q_k) + dwBw(q_k)]/2. \quad (2)$$

For each ranked task M_i , the algorithm uses the set mBW to calculate the bandwidth $BandRate$ on all the set of Average bandwidth $Mean_{Bw}(q_k)$, then it will scan all the resources to find the node q_k that minimizes $BandRate$ of the task M_i (see Eq(3)):

$$BandRate(M_i, q_k) = mBW(M_i)/Mean_{Bw}(q_k). \quad (3)$$

3.3.3 Algorithm

Algorithm 1 represents the Bw-AWA Algorithm, which allocates each ranked task respecting the priority order and chooses the best resource to execute the current task in terms of bandwidth metrics.

Algorithm 1.

Input: \mathcal{TG} (Tasks Graph), \mathcal{RG} (Resources Graph).

Output: A mapped tasks to optimal resources.

Initialisation: $Order_{list} = \emptyset, Mean(q_k) = \emptyset, bandRate(M_i, q_k) = \emptyset$.

Read $Com_c(M_i)$ and $T_{cc}(M_i)$

Calculate $rank(M_i)$ for all tasks (see Algorithm 2)

Order tasks: Place tasks in descending order in $Order_{list}$

While non-placed task in $Order_{list}$ {

Select the first task from the $Order_{list}$

$q = q_1$

Assign task M_i *to the resource* q

For Each $q_k \in Q - \{q_1\}$

{

$Mean(q_k) = [upBw(q_k) + dwBw(q_k)]/2$

$bandRate(M_i, q_k) = mBW(q_k)/Mean(q_k)$

IF $bandRate(M_i, q_k) < bandRate(M_i, q_{k-1})$ *and* q_k *is Empty*

{ $q = q_k$

Assign current task M_i *to the resource* q_k }

}

Remove task M_i *from* $Order_{list}$ }

The ranking phase is represented in the Algorithm 2, with the aim to order the tasks according to the rank calculated in descending order.

Algorithm 2.

Input: $Com_c(M_i)$, $T_{cc}(M_i)$
Output: *Ranked tasks*
Initialisation: $M_i = Task_{exit}$
 $Rank(M_i) = Com_c(M_i) + \min_{M_j \in pred(M_i)} (T_{cc}(M_j) + Rank(M_j))$

4 Use Case and Experimental Results

4.1 Workflow System for Smart Monitoring

This scenario is based on a dispersed network of security, manufacturing, transportation, and healthcare surveillance cameras [27].

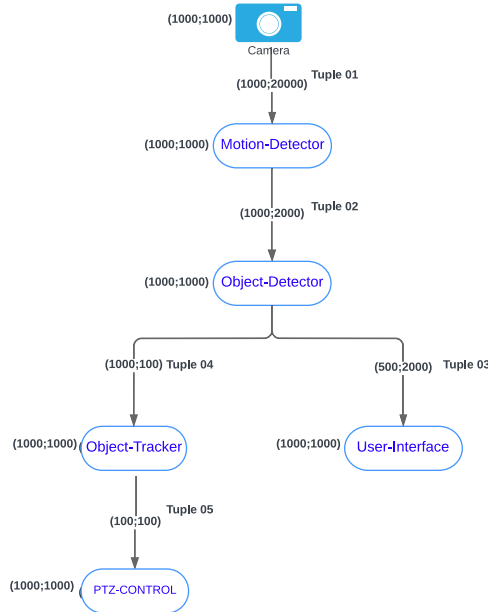


Figure 2. A Monitoring Workflow Graph

The tasks are dependent on the positioning of an IoT application in the Fog-Cloud environment. The application consists of five tasks, in which each Fog device will be in charge of a security camera number. Using the raw video stream (Tuple 01), the Motion-Detector locates any motion. The motion video stream (Tuple 02) is then sent on to the Object-Detector, which compares the images and finds the moving object. Following moving object recognition, the discovered object is delivered to the User-Interface (Tuple 03) or to an Object Tracker for location monitoring (Tuple 04). The Object-Tracker sends a location to the PTZ (Pan Tilt Zoom)-Control (Tuple 05), which serves as an actuator to turn the camera there. Tasks and Edges have two main attributes (processing requirements and bandwidth requirements) respectively (see Figure 2).

The system consists of IoT Devices, gateways (Fog nodes), and a Cloud platform. The system configuration parameters used for the monitoring case are presented in Table 2 [27]. The latency between

Table 2. Configuration Details in the System

Type	MIPS	RAM	Up BW	Link	Down Link BW
Cloud	44800	40000	100		10000
Proxy	2800	4000	10000		10000
Gateway	2800	4000	10000		10000
IoT devices	500	1000	10000		10000

network resources is 2, 4, and 100 (ms) respectively between [IoT Devices-Gateway], [Gateway-Proxy], and [Proxy-Cloud]. iFogSim simulator [27] is used to simulate the proposed scenario. Table 3 represents the task ranking phase (see Eq (1)) with the priority order.

The MIPS Rate ($MIPS_{Rate}$) values of the application tasks in different network resources are calculated using Eq (4) and presented in Table 4:

$$MIPS_{Rate} = MIPS_{Task} / MIPS_{resource}. \quad (4)$$

This parameter is used in HEFT and CPOP algorithms in the task allocation phase.

Table 3. Task ranking

Task	ranking	order
motion detector	7000	1
object detector	3000	2
object tracker	1000	3
user interface	1000	4

Table 4. MIPS Rate Of All Tasks In different network resources

Network Resources	$MIPS_{Rate}$
CLOUD Device	0.02336
Proxy	0.3571
Gateway	0.3571
CAMERA Devices	2.0

Table 5 contains two main parameters used in our proposed algorithm. The first one is the bandwidth Mean of different kinds of resources (see Eq 2). The bandwidth Mean result is used to calculate the second parameter which is the BandRate (see Eq 3).

Table 5. Bandwidth Rate Of All Tasks In different network resources

resource	$Mean_{Bw}$	$bandRate$
CLOUD Device	5050.0	0.198
Proxy	10000.0	0.1
Gateway	10000.0	0.1
Camera Devices	10000.0	0.1

4.2 Experimental Results

4.2.1 Variation in Number of Fog Nodes and Cameras

In our case study, the application has three different settings:

1. The simulation is based on four configurations: (1,4,7), (2,4,12), (3,4,17), and (4,4,22). Each configuration contains three parameters representing, respectively, Fog devices, camera devices, and the total resources including one proxy and one Cloud server.
2. The number of IoT devices (refer to the camera devices) related to each Fog node is augmented to 8,10,12,16.
3. The number of Fog nodes is augmented to 10,20,30,40 nodes.

4.2.2 Makespan

This is the value given by the total execution time. It is calculated based on the simulation start time and the simulation end time. The simulator's executive clock is used to calculate this value in milliseconds [28](see Eq 5):

$$Makespan = \sum (finish(Mi)) - \sum (start(Mi)), \quad (5)$$

where the function $finish(Mi)$ gives the completion time related to the last task in the workflow and $start(Mi)$ gives the start time of the first task scheduled. Our work is compared to HEFT, CPOP [12], Popularity [24], and FCFS [15] strategies. Firstly, we compare BW-AWA with HEFT and CPOP well-known priority scheduling algorithms that aim to reduce the Makespan for scheduled tasks. The approach HEFT is used as an algorithm in the Cloud-Fog-IoT infrastructure [29], it is a placement algorithm for heterogeneous systems. However, in Bw-AWA, the determination of priority is also related to task ranking but the resource allocation deals with the rated bandwidth of each task in all processing nodes to select the optimal processing node. In the algorithm [24], the tasks are placed along the shortest network path between the user and the cloud provider, and priority should be given when allocating an application's interconnected tasks. The topological arrangement of the tasks determined the placement order, putting the initial tasks with the pre-fixed priority nearer to the Fog devices. The figures below show the impact of changing the number of resources on Makespan (milliseconds). In the graphics, we used a logarithmic scale on the Y-axis to highlight the differences between the strategies. On the X-axis, we have taken four configurations. The time taken

by an algorithm for the execution of the tasks is crucial in choosing the algorithm. Figures 3, 4, and 5 demonstrate the simulation results graphs in varied configurations.

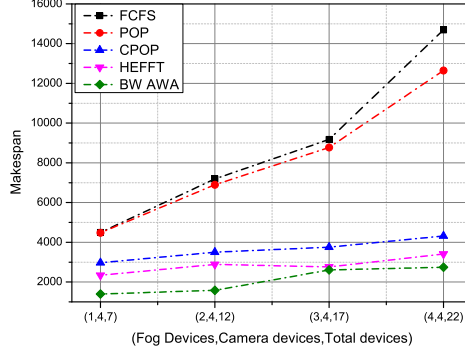


Figure 3. Makespan Results (First Configuration)

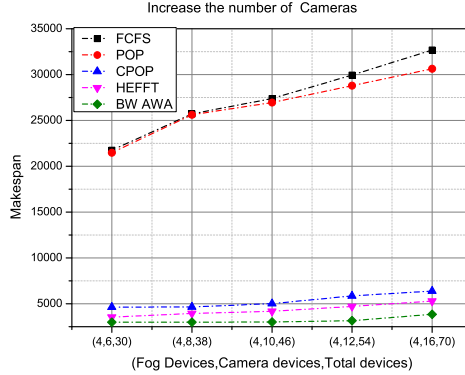


Figure 4. Increase IoT Devices Number

Differences among the five algorithms are shown in Tables 6 and 7. The BW-AWA algorithm results outperform other approaches for the following reasons:

Firstly, BW-AWA leverages Fog devices as processing nodes for allocating resources. When the devices number increases, the Makespan

Table 6. Makespan results

Figure	Config	FCFS	Popularity	HEFT	CPOP	BW-AWA
Figure 3	(1,4,7)	4494	4482	2340	2973	1396
	(2,4,12)	7193	6894	2886	3502	1580
	(3,4,17)	9179	8772	2758	3750	2607
	(4,4,22)	14694	12644	3408	4319	2746
Figure 4	(4,6,30)	21756	21483	3545	4629	2984
	(4,8,38)	25712	25624	3945	4661	2998
	(4,10,46)	27394	26945	4170	5024	3005
	(4,12,54)	29942	28792	4716	5855	3153
	(4,16,70)	32691	30633	5292	6382	3858
Figure 5	(10,4,52)	28558	20732	5022	8790	3195
	(20,4,102)	47912	41405	7566	9354	5645
	(30,4,152)	71519	66949	11219	11683	8003
	(40,4,202)	94456	85983	13239	14597	10714

Table 7. Improvement ratio of Makespan of the proposed solution

Figure	Config	FCFS	Popularity	HEFT	CPOP
Figure 3	(1,4,7)	68,94%	68,85%	40,34%	53,04%
	(2,4,12)	78,03%	77,08%	45,25%	54,88%
	(3,4,17)	71,60%	70,28%	5,47%	30,48%
	(4,4,22)	81,31 %	78,28%	19,42%	36,42%
Figure 4	(4,6,30)	86,28 %	86,11%	15,83 %	35,54%
	(4,8,38)	88,34 %	88,30 %	24,01 %	35,68%
	(4,10,46)	89,03 %	88,85 %	27,94 %	40,19%
	(4,12,54)	89,47 %	89,05 %	33,14 %	46,15%
	(4,16,70)	88,20 %	87,41 %	27,10 %	39,55%
Figure 5	(10,4,52)	88,81%	84,59 %	36,38 %	63,65 %
	(20,4,102)	88,22 %	86,37 %	25,39 %	39,65 %
	(30,4,152)	88,81 %	88,05 %	28,67 %	31,50%
	(40,4,202)	88,66 %	87,54%	19,07 %	26,60 %

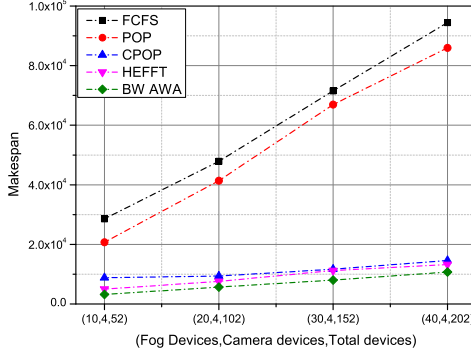


Figure 5. Increase Fog nodes Number

for all approaches increases, and it always remains reduced compared to the other four approaches. Secondly, according to Table 5, BW-AWA considers the task's band rate, which is resided in the gateway devices. As a result, extra communication time to the Cloud nodes is reduced. Because the resources with huge bandwidth could finish their work quickly, other resources with lower Bandwidth would still operate for a longer period of time when the disparity between resources is large. So, the proposed algorithm saves time more when there are more Fog devices.

HEFT and CPOP are almost similar in that they rely on processing for placement. These processing algorithms will tend to place on powerful nodes according to the MIPS Rate values in Table 4. In this case, we will have massive use of the Cloud, which has a high latency of 100 ms, but it is necessary to consider nearby resources in the allocation phase. Although CPOP and HEFT have the same processing complexity, CPOP processing time exceeds that of HEFT. Because it performs another pass to calculate rank downward tasks, it tries to reduce the critical path impact of the graph.

FCFS method processes tasks in the order they are entered, regardless of the length or size of the task. It is difficult to reduce the scheduling time. The results show that as the number of IoT devices increases, the Makespan increases. Popularity presents a priority rule

in which the most popular tasks are placed first in the devices closest to the end-users. Furthermore, Popularity assigns also to upper levels despite the Fog device's capacities and the task's requirements. Popular tasks had reduced latency, while less popular ones saw greater lag.

4.2.3 Throughput

Figures 6,7, and 8 show a comparison of throughput rates. The total number of tasks completed successfully in a certain time period is referred to as Throughput [30].

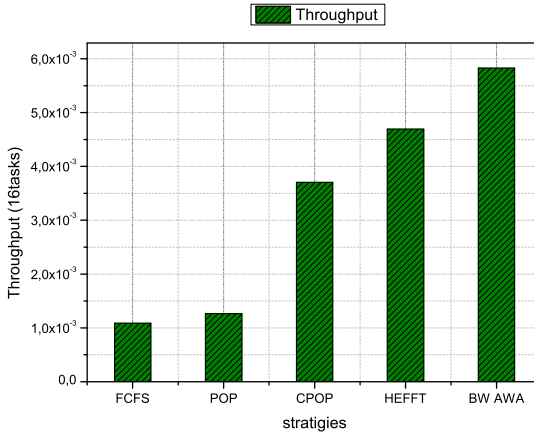


Figure 6. 16 Tasks

It is proportional to the number of tasks locally and remotely allocated based on the number of camera devices generating the task.

The effectiveness of scheduling approaches is demonstrated by using Eq 6:

$$Throughput = Tasks\ Number / Makespan. \quad (6)$$

The findings show that, when the task number rises, the throughput rate rises as well. When a rate is higher, it improves the execution of the tasks and it also reduces the makespan [30]. The throughput of our proposed approach is compared to that of the other chosen techniques. As a result, it provides better results. Three configurations, with 16,40,

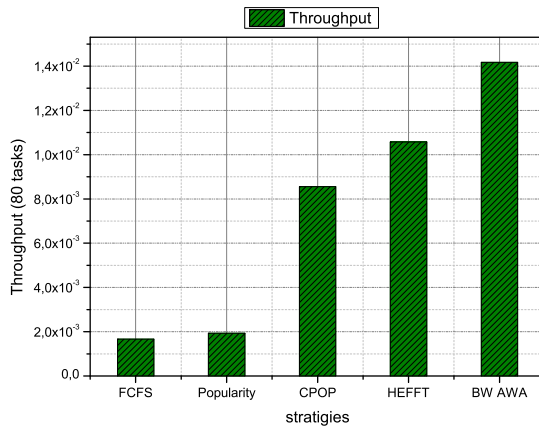


Figure 7. (80 Tasks)

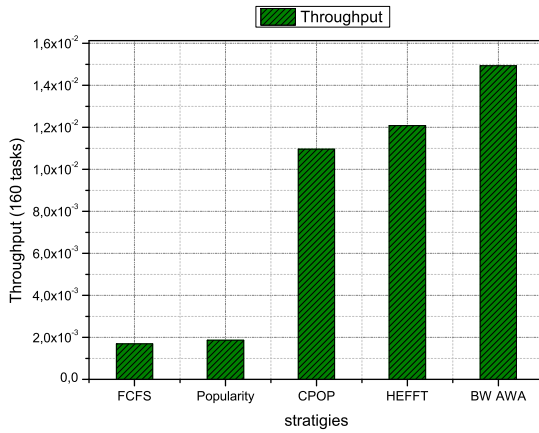


Figure 8. (160 Tasks)

and 160 tasks, respectively, were used to analyze throughput. As expected, the results show that the throughput increases as the number of processing nodes increases (202 total resources used in 160 tasks). Along with this, as the number of tasks increases, the throughput also increases.

We can conclude that the proposed algorithm BW-AWA can increase task scheduling effectiveness based on the experimental results mentioned above. This algorithm can therefore enable to reduce the makespan in the Fog-Cloud computing system.

5 Conclusion and Future Work

Fog Computing is an emerging computing paradigm that supports and unleashes the full reach of IoT and solves the limitations of the Cloud. In order to improve the processing node's efficiency and optimize performance indicators, the optimizing task scheduling problem in a Fog-Cloud environment has taken a lot of intentions by researchers. In this paper, we have dealt with the problem of workflow application task scheduling in a Fog-Cloud environment, focusing on the makespan optimization which is the workflow execution time. The proposed list scheduling algorithm (BW-AWA) takes into consideration task priority, communication cost between tasks, and processor choice in terms of bandwidth over the entire schedule. The comparative study between the results obtained from our approach and those of the selected algorithms clearly shows that our proposed algorithm provides efficient and effective solutions in terms of Makespan with the task priority constraint. In order to improve our approach, we intend in our future work to improve our algorithm to support other QoS (Quality of Service) metrics, such as budget constraints and power consumption through the processor Dynamic Voltage and Frequency Scaling (DVFS) technique, and subsequently formally move from a single-objective optimization problem to a multi-objective optimization problem.

References

- [1] V. G. Skobelev and V. V. Skobelev, "A general analytical model of queuing system for Internet of Things applications," *Computer*

- Science Journal of Moldova*, vol. 28, no 1(82), pp. 3–20, 2020.
- [2] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, “Zenith: Utility-Aware Resource Allocation for Edge Computing,” in *2017 IEEE International Conference on Edge Computing (EDGE)*. Honolulu, HI, USA: IEEE, 2017, pp. 47–54. DOI: 10.1109/IEEE.EDGE.2017.15.
 - [3] J. K. Zao et al., “Augmented Brain Computer Interaction Based on Fog Computing and Linked Data,” in *2014 International Conference on Intelligent Environments*. Shanghai, China: IEEE, 2014, pp. 374–377. DOI: 10.1109/IE.2014.54.
 - [4] N. Verba et al., “Modeling industry 4.0 based Fog computing environments for application analysis and deployment,” *Future Generation Computer Systems*, vol. 91, pp. 48–60, 2019.
 - [5] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, “Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4548–4556, Oct. 2018. DOI: 10.1109/TII.2018.2818932.
 - [6] R. Yayla, E. Albayrak, U. Yüzgeç, “Vehicle Detection from Unmanned Aerial Images with Deep Mask R-CNN,” *Computer Science Journal of Moldova*, vol. 30, no 2(89), pp. 148–169, 2022.
 - [7] D. Ye, M. Wu, S. Tang, and R. Yu, “Scalable Fog Computing with Service Offloading in Bus Networks,” in *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*. Beijing, China: IEEE, 2016, pp. 247–251. DOI: 10.1109/CSCloud.2016.34.
 - [8] X. Wang, Z. Ning, and L. Wang, “Offloading in Internet of Vehicles: A Fog-Enabled Real-Time Traffic Management System,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018. DOI: 10.1109/TII.2018.2816590.
 - [9] R. K. Lomotey, J. Pry, S. Sriramoju, “Wearable IoT data stream traceability in a distributed health information system,” *Pervasive and Mobile Computing*, vol. 40, pp. 692–707, 2017.
 - [10] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, “Optimal Joint Scheduling and Cloud Offloading for Mobile Applications,” *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 301–313, 2019. DOI: 10.1109/TCC.2016.2560808.

- [11] S. Sharma, H. Saini, “A novel four-tier architecture for delay aware scheduling and load balancing in Fog environment,” *Sustainable Computing: Informatics and Systems*, vol. 24, Article no. 100355, 2019.
- [12] H. Topcuoglu, S. Hariri, and Min-You 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, March 2002. DOI: 10.1109/71.993206.
- [13] A. Agarwal and P. Kumar, “Economical Duplication Based Task Scheduling for Heterogeneous and Homogeneous Computing Systems,” in *2009 IEEE International Advance Computing Conference*. Patiala, India: IEEE, 2009, pp. 87–93. DOI: 10.1109/IADCC.2009.4808986.
- [14] X. Tang, K. Li, G. Liao, R. Li, “List scheduling with duplication for heterogeneous computing systems,” *Journal of parallel and distributed computing*, vol. 70, no. 4, pp. 323–329, 2010.
- [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. DOI: 10.1109/MCC.2017.27.
- [16] P. Varshney and Y. Simmhan, “Characterizing application scheduling on edge, fog, and cloud computing resources,” *Software: Practice and Experience*, vol. 50, no. 5, pp. 558–595, 2020.
- [17] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, “Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System,” *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016. DOI: 10.1109/TC.2016.2536019.
- [18] S. Bitam, S. Zeadally, and A. Mellouk, “Fog computing job scheduling optimization based on bees swarm,” *Enterprise Information Systems*, vol. 12, no. 4, pp. 373–397, 2018.
- [19] B. M. Nguyen, H. T. T. Binh, T. T. Anh, and D. B. 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, Article no. 1730, 2019. <https://doi.org/10.3390/app9091730>.

- [20] D. Rahbari and M. Nickray, "Scheduling of fog networks with optimized knapsack by symbiotic organisms search," in *2017 21st Conference of Open Innovations Association (FRUCT)*. Helsinki, Finland: IEEE, 2017, pp. 278–283. DOI: 10.23919/FRUCT.2017.8250193.
- [21] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, H. Ijaz, "A job scheduling algorithm for delay and performance optimization in Fog computing," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, Article ID. e5581, 2020.
- [22] Y. Wang, C. Guo, and J. Yu, "Immune scheduling network based method for task scheduling in decentralized Fog computing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID. 2734219, 2018. <https://doi.org/10.1155/2018/2734219>.
- [23] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. Kanazawa, Japan: IEEE, 2016, pp. 1–4. DOI: 10.1109/APNOMS.2016.7737240.
- [24] C. Guerrero, I. Lera, and C. Juiz, "A lightweight decentralized service placement policy for performance optimization in Fog computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 6, pp. 2435–2452, 2019.
- [25] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "How to place your apps in the Fog: State of the art and open challenges," *Software: Practice and Experience*, vol. 50, no. 5, pp. 719–740, 2020.
- [26] M. Sulaiman, Z. Halim, M. Waqas, and Doğan Aydın, "A hybrid list-based task scheduling scheme for heterogeneous computing," *The Journal of Supercomputing*, vol. 77, pp. 10252–10288, 2021.
- [27] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, vol. 47, no 9, pp. 1275–1296, 2017.
- [28] L. Hamid, A. Jadoon, and H. Asghar, "Comparative analysis of task level heuristic scheduling algorithms in Cloud computing," *The Journal of Supercomputing*, vol. 78, pp. 12931–12949, 2022.

- [29] X.-Q. Pham, N. D. Man, N. D. T. Tri, N. Q. Thai, and E.-nam Huh, “A cost-and performance-effective approach for task scheduling based on collaboration between Cloud and Fog computing,” *International Journal of Distributed Sensor Networks*, vol. 13, no. 11, Article no. 1550147717742073, 2017.
- [30] M. Aljarah, M. Shurman, and S. Alnabelsi, “Cooperative hierarchical based Edge-computing approach for resources allocation of distributed mobile and IoT applications,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no 1, pp. 296–307, 2020.

Bentabet Dougani, Abdeslem Dennai,

Received December 24, 2022

Revised January 25, 2023

Accepted January 28, 2023

Bentabet Dougani

ORCID: <https://orcid.org/0000-0003-4295-3830>

CCAI Research Team, SGRE Laboratory,

University of BECHAR, Algeria

E-mail: dougani.bentabet@univ-bechar.dz

douganibentabetinfo@gmail.com

Abdeslem Dennai

ORCID: <https://orcid.org/0000-0003-2133-0765>

CCAI Research Team Leader, SGRE Laboratory,

University of BECHAR, Algeria

E-mail: dennai.abdeslam@univ-bechar.dz