International Journal of Electrical and Computer Engineering (IJECE)
Vol. 14, No. 4, August 2024, pp. 4445~4455
ISSN: 2088-8708, DOI: 10.11591/ijece.v14i4.pp4445-4455

Efficient offloading and task scheduling in internet of things-cloud-fog environment

Marwa Gamal[1], Samar Awad[1], Rehab F. Abdel-Kader[2], Khaled Abd El Salam[1]
[1]Electrical Engineering Department, Faculty of Engineering, Suez Canal University, Ismailia, Egypt
[2]Electrical Engineering Department, Faculty of Engineering, Port Said University, Port Said, Egypt

Article Info

Article history:

Received Nov 16, 2023
Revised Apr 25, 2024
Accepted May 12, 2024

Keywords:

Cloud computing
Fog computing
Offloading strategy
Scientific workflow
Task scheduling

ABSTRACT

Efficient offloading and scientific task scheduling are crucial for managing computational tasks in research environments. This involves determining the optimal location for executing a workflow task and allocating the task to computing resources to optimize performance. The challenge is to minimize completion time, energy consumption, and cost. This study proposes three methods: latency-centric offloading (LCO) for delay-sensitive applications; energy-based offloading (EBO) for energy-saving; and efficient offloading (EO) for balanced task distribution across tiers. Scheduling in this paper uses a genetic algorithm (GA) with a weighted sum objective function considering makespan, cost, and energy for internet of things-cloud-fog (IoT-fog-cloud). Comparative studies involving montage, Cybershake, and epigenomics workflows indicate that LCO excels in terms of makespan and cost but ranks the lowest in energy. EBO excels in energy efficiency, aligning closely with the base method. EO competes effectively with the base method in terms of makespan and cost but consumes more energy. This research enables the selection of the most suitable method based on the type of application and its prioritization of makespan, energy, or cost.

Corresponding Author:

Samar Awad
Department of Electrical Engineering, Faculty of Engineering, Suez Canal University
Alfrosia street, Ismalia, Egypt
Email: Samar_Awad@eng.suez.edu.eg

1. INTRODUCTION

A scientific workflow outlines a process to achieve a scientific goal, defined by tasks and their interdependencies [1]. Dependencies occur at various stages, directing tasks to be executed sequentially to achieve the scientific goal [2]. Scientific workflows commonly use directed acyclic graphs (DAGs) to model these dependencies, with tasks as nodes and dependencies as edges [3]. The evolution of computationally and data-intensive methods in the natural sciences has driven the creation of scientific workflows designed to automate repetitive computational tasks [4]. Initially, scientific workflows were primarily deployed on distributed systems [5], [6] and on high-performance computing (HPC) [7]. During this period, the focus centered around treating systems and applications as opaque entities, emphasizing distributed resource management and workload execution [6]. Recently, there has been a shift toward using cloud computing infrastructure to conduct scientific workflows [8]–[10]. Unlike distributed systems, cloud computing operates on a client-server architecture, centrally utilizing resources with a pay-as-you-go model [11].

Cloud solutions may not consistently meet quality of service (QoS) and quality of experience (QoE) for certain latency-sensitive internet of things (IoT) applications due to distance and connectivity issues. This led to fog computing, extending cloud resources closer to IoT devices. Fog devices process and offload functions from cloud servers, ensuring enhanced performance [12]. Advancements in networking technology,

including 5G and beyond, have generated a growing demand for extensive computing and seamless service access due to the proliferation of mobile devices and Internet-connected users. This has led to the emergence of fog radio access networks (F-RANs) [13] in conjunction with cloud radio access networks (C-RANs) [14], collectively contributing to pervasive computing services. F-RANs position a fog computing layer at the network's edge, allowing local handling of some services and applications, eliminating the need for centralized cloud computing. This method improves how F-RANs handle different quality needs in 5G services by making distributed caching and centralized processing more efficient. Fog computing not only delivers cloud-like services to end users (EUs) but also has the potential to enhance the performance of fog-based systems, improving system efficiency, reducing service delays [15], and reducing cost [16] through effective computational offloading algorithms [17].

Combining the internet of things (IoT), fog computing, and cloud technologies creates IoT-fog-cloud (IFC) systems. These systems provide seamless services and applications, leading to improved QoS across a range of devices, from edge devices to cloud resources. This ensures the efficient offloading of computations. Assigning workflow tasks to virtual machines in the cloud is a crucial step in scheduling workflow execution on cloud resources. However, the high costs of computation and communication often pose challenges [18], [19].

To address challenges in scheduling jobs on cloud infrastructure, population-based methods like genetic algorithms (GA) and particle swarm optimization (PSO) have been proposed as solutions. Typically, these methods follow a predefined mapping process, falling under the category of static workflow scheduling mechanisms. The main optimization objectives include maximizing completion execution time, often referred to as "makespan", and minimizing costs. Furthermore, these techniques often incorporate other QoS criteria, such as budget constraints and deadlines, as integral aspects of the optimization process.

Recent studies focusing on scheduling scientific workflows show that the GA is the top choice for optimization based on population in this field [20]–[22]. The GA is widely used to solve task scheduling problems [23] due to its global optimization and search ability. This paper employs a three-tier network architecture. An initial iteration of this research was presented and published at [24]. Three offloading strategies are proposed in [25]. Subramoney and Nyirenda [26] introduced a method called multi-swarm particle swarm optimization (MS-PSO) to enhance the scheduling of scientific tasks in IoT-cloud-fog systems.

The main contribution of this paper is described as proposing of latency-centric offloading (LCO), energy-based offloading (EBO), and efficient offloading (EO) as offloading strategies with consideration of real-time applications using GA in [26] for scheduling scientific workflows, along with their practical realization within the FogWorkflowSim framework [25]. A comparative analysis of offloading strategies, including LCO, EBO, EO, and the base method, has been carried out for scientific workflows. This evaluation encompasses performance metrics such as makespan, cost, and energy. In our findings, by employing one of the proposed methods, we achieved improved metrics for various applications. For instance, LCO is optimal for latency-sensitive applications, EBO excels in energy reduction scenarios, and EO is preferable for applications seeking a balance across all metrics.

The subsequent sections of this paper are structured as follows: section 2 formulates the proposed methods as problem definition, suggested offloading strategies, and an explanation of the task scheduling concept. Section 3 presents the different workflow models, the simulated environment, and the performance assessment. Ultimately, section 4 offers concluding remarks for the paper and suggested future work.

2. METHOD

IoT-fog-cloud offloading optimizes task execution in a multi-layered architecture. It involves deciding whether to execute tasks locally, offload to a nearby fog node, or process in the cloud. Our study, considering real-time applications and diverse needs, aims to optimize factors like latency, energy efficiency, and system performance. We propose three offloading strategies and compare them with existing ones in a workflow simulator [25].

2.1. Problem definition

The configuration described in this study involves an IoT-fog-cloud environment arrangement. The primary focus lies in effectively orchestrating the controller's efforts to locate the optimal server for various tasks. This necessitates the establishment of criteria for assessing the effectiveness of task offloading. Subsequently, we proceed to develop a cost function for offloaded tasks, considering both the time delay and energy consumption. Firstly, we calculate the time to offload to the cloud, fog, and local execution time (non-offloading time at the end device) as T_C, T_{F}, and T_{ED}, respectively. Then calculate the energy required to offload to cloud, fog, and local execution as E_C, E_F, and E_{ED}, respectively. To begin, let's initiate our discussion with a delay analysis, as in [27].

$$T_{\mathrm{ED}} = \frac{D}{R} \quad (1)$$

$$T_{\mathrm{F}} = T_{\mathrm{ED}} + \frac{M}{S_f} \quad (2)$$

$$\mathrm{T_C} = T_{\mathrm{ED}} + \frac{M}{S_c} \quad (3)$$

where D represents the size of the input bits and R denotes the processing rate, S_f refers to the fog server's computation capacity, which represents the ratio between the tuning parameter and the bandwidth between the mobile device and fog server, and S_c stands for the capacity of computation in the cloud server. Then we analyze the energy consumption.

$$E_{ED} = T_{ED} * p_{mc} \quad (4)$$

where p_{mc} refers to the maximum amount of power expended by the mobile device during its operation or task execution.

$$E_F = E_{F1} + E_{F2} \quad (5)$$

Here, E_{F1} denotes the idle energy consumption of the mobile device, and E_{F2} represents the energy consumed by the mobile device for transmitting the input bits to the fog server.

$$E_{\mathrm{F1}} = \frac{Mp_i}{S_f} \quad (6)$$

$$E_{\mathrm{F2}} = \frac{Mp_{tr}}{R} \quad (7)$$

Here, p_i represents the idle power of the mobile device, p_{tr} signifies the transmission power of the mobile device, and M indicates the size of the instructions executed on the fog server.

$$E_C = E_{C1} + E_{C2} \quad (8)$$

where E_{C1} is the idle energy consumption of the mobile device, and E_{C2} represents the energy consumed by the mobile device for transmitting the input bits to the cloud server.

$$E_{\mathrm{C1}} = \frac{Mp_i}{S_c} \quad (9)$$

$$E_{\mathrm{C2}} = \frac{Mp_{tr}}{R} \quad (10)$$

where M indicates the size of the instructions executed on the cloud server. The aim of our proposed work is to use these parameters to make an efficient decision about which tasks should be processed at the fog nodes, which should be sent to the cloud, and which should be executed locally at end devices, considering various factors such as computational capacity, network conditions, energy constraints, and user requirements.

2.2. The proposed methods

Our proposed method is developed by building upon concepts from base strategies and incorporating ideas from [25], [27], where the offloading algorithm is formulated as an optimization approach to reduce total execution time, total cost, and energy consumption within specified time constraints. The decision on offloading is influenced by factors like application type, task characteristics, and resource capacities. Hence, our aim is to devise methods capable of addressing these diverse scenarios, particularly those suitable for real-time applications.

2.2.1. Latency centric offloading

Offloading decisions prioritize minimizing task completion time (latency). Tasks may be offloaded to fog nodes or the cloud to meet latency requirements, or they may be executed locally with no offload. This strategy finds application in various real-world scenarios where low latency is critical. Algorithm 1 illustrates the LCO strategy. The workflow wf is the input to the algorithm, J jobs, related deadlines, and N devices,

while the output is the offloading decision. We assume a numeric representation for the offloading decision, as 0 denotes non-offloading (locally execution), 1 represents offloading to fog, and 2 stands for offloading to cloud.

Algorithm 1. Latency-centric offloading LCO strategy

```
Input: Workflow wf composed of J jobs, related deadline, and N devices;
Output: Offloading decision;
Initialize LAN Bandwidth, WAN Bandwidth, and Tuning parameter;
for i ← 1, N do
     Calculate T_ED;
     Calculate T_F;
     Calculate T_C;
  if deadline < min (T_C, T_F, T_ED) then
     Offload decision ⟵ 0
 else if deadline > min (T_C, T_F, T_ED) &&
     min (T_C, T_F, T_ED) = T_F then
     Offload decision ⟵ 1
 else if deadline > min (T_C, T_F, T_ED) &&
     min (T_C, T_F, T_ED) = T_C then
     Offload decision ⟵ 2
 else
     Offload decision ⟵ 0
```

2.2.2. Energy based offloading

It is idea is close to the simple method. Offloading decisions may be made with the aim of conserving energy on IoT devices. For example, tasks that consume a lot of energy may be offloaded to more powerful fog nodes or the cloud. It is useful to identify scenarios where computational resources are constrained and where the optimization of energy consumption is critical. Algorithm 2 formulates the EBO strategy. The workflow wf is the input to the algorithm, J jobs, related deadlines, and N devices, while the output is the offloading decision.

Algorithm 2. Energy-based offloading EBO strategy

```
Input: Workflow wf composed of J jobs, related deadline, and N devices;
Output: Offloading decision;
Initialize LAN Bandwidth, WAN Bandwidth, and Tuning parameter;
for i ← 1, N do
     Calculate T_ED;
     Calculate T_F;
     Calculate T_C;
  if deadline < min (T_C, T_F, T_ED) then
     Offload decision ⟵ 0
 else if min (E_C, E_F, E_ED)== E_F then
     Offload decision ⟵ 1
 else if min (E_C, E_F, E_ED)== E_C then
     Offload decision ⟵2
 else
     Offload decision ⟵ 0
```

2.2.3. Efficient offloading

Leverage hybrid IoT-fog-cloud architectures to exploit cost differentials between different resource types. Delegate tasks to the resource that strikes the optimal balance between time and energy considerations. In the simple offloading strategy outlined in [25], the offloading decision process is initially influenced by the job's deadline constraint. It then weighs the trade-off between job execution times across the three tiers, followed by an assessment of energy consumption. The task is offloaded to the tier that both meets the deadline constraint and offers the shortest execution time while also minimizing energy consumption.

Our findings indicate that the decision-making process prioritizes implementation time before considering energy consumption. This led to certain jobs being chosen based on time considerations, despite minimal discrepancies across different layers. In contrast, there were substantial disparities in energy consumption. Hence, in this approach, we sought to mitigate this limitation by striking a balance between time and energy considerations. We computed the normalized values for both time and energy consumption and then integrated these two ratios into a fitness function. This enabled us to arrive at a decision that considers both factors.

$$T_{\text{norm}} = \frac{T_i - T_{min}}{T_{max} - T_{min}}$$
$$E_{\text{norm}} = \frac{E_I - E_{min}}{E_{max} - E_{min}} \quad (11)$$

where T_{norm} and E_{norm} are normalized execution time and energy consumption. T_i and E_I refer to the execution times and energy consumed by i^{th} job, respectively. The terms min and max represent the minimum and maximum of both execution time and energy consumption when ith job is executed in different three tiers. Calculate the fitness function depending on time and energy components as (12).

$$fitness = T_{norm} + E_{norm} \quad (12)$$

The job is offloaded to the tier that meets the minimum fitness standards. Algorithm 3 formulates the EO strategy. The workflow wf is the input to the algorithm, J jobs, related deadlines, and N devices, while the output is the offloading decision. Comparing with one of the methods available in [26], which is selecting the simple method in [25] as an offloading with GA as a scheduling algorithm, we found that, according to the application's nature and using one of the methods proposed in this research, we achieved better ratios for all the different metrics. For example, LCO will be better when using applications that are more sensitive to latency, EBO will be better when using applications that concern energy reduction, and EO will be better when using applications that need to achieve a balance between all metrics.

Algorithm 3. Efficient offloading EO strategy

```
1. Input: Workflow wf composed of J jobs, related deadline, and N devices;
2. Output: Offloading decision;
3. Initialize LAN Bandwidth, WAN Bandwidth, and Tuning parameter;
4. for i ← 1, N do
5.      Calculates T_ED, E_ED, T_F, E_F T_C, and E_C;
7.   if deadline < min (T_C, T_F, T_ED) then
8.      Calculate T_max=max (T_C,T_F,T_ED), T_min=min (T_C,T_F,T_ED);
9.      Calculate E_max=max (E_C,E_F,E_ED), E_min=min (E_C,E_F,E_ED);
10.     Calculate T_norm and E_norm for job in different tiers;
11.     Calculate fitness for each tiers;
12.     Job is offloaded to tier that fulfils the minimum fitness;
13.  Else
14.     The job is executed locally.
```

2.3. Task scheduling using GA and proposed objective function

Task scheduling algorithms must adapt to these offloading decisions to ensure optimal resource allocation and task execution. So, this research uses GA, as in [26], for the task scheduling process. This study optimizes task offloading and resource selection with three key objectives: minimize makespan, reduce costs, and optimize energy consumption. The workflow makespan is denoted as the cumulative execution time required for successful completion; the total cost includes both communication and computation expenses, and energy consumption is established using idle and active components. The task scheduling process aims to employ the three stated objectives by using a weighted sum objective function, which is formulated as (13):

$$F(p) = w_1.MS + w_2.TC + w_3\,.E \quad (13)$$

In the context of a GA, the allocation from a workflow to the available computing resources of n tasks are represented by p, termed a chromosome. Parameters MS, TC, and E correspond to values of makespan, total cost, and energy consumption, respectively. The coefficient weight $w*$, with equal weights ($w* = 0.2$), is used to ensure an equal contribution from each objective in performance assessments.

2.3.1. Assignment of workflow tasks to computational resources

In this study, workflows are scheduled to run on the source (end device), a fog virtual machine (VM), or a cloud VM. End devices exclusively offload tasks to cloud and fog resources. Workflow scheduling involves a single representative end device in the encoding process. Natural numbers identify individuals for GA, representing task-resource schedule mappings (chromosomes). Each chromosome p, of length n, corresponds to the total number of tasks. Each position in the chromosome is a positive integer denoting the task number, with the assigned value indicating the VM ID for task execution. VM ID numbers are chosen from available VMs in the respective tier. For instance, in Figure 1, an example workflow with 10 tasks maps to an IoT-cloud-fog setup with 3 cloud VMs and 2 fog VMs. In this example, p = (3, 5, 2, 4, 1, 3, 2, 5) represents the chromosome.

2.3.2. The process of optimization

In the GA approach, a sequence of operations, including selection, crossover, and mutation, generates a new set of potential solutions. These solutions undergo assessment via workflow simulation,

producing their respective fitness values. The chromosome with the best fitness, along with its associated value, is preserved, employing elitism to retain the best solutions for the next iteration.

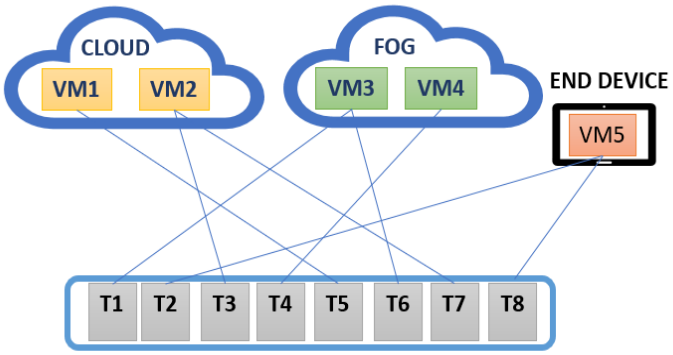

Figure 1. Example for Workflow scheduling model

3. RESULTS AND DISCUSION

This section commences with a detailed overview of various workflow models, emphasizing their significance in the study's context. It then meticulously outlines the simulation environment setup, leveraging the FogWorkflowSim Toolkit [25] for precision. Subsequently, it presents experimental results, providing valuable insights into the performance and dynamics of simulated workflow models, followed by a comprehensive discussion to deepen the reader's understanding.

3.1. Workflow models

This study utilizes three well-established scientific workflow applications: Montage, Epigenomics, and CyberShake, as detailed in [28]. Montage generates customized sky mosaics for astronomy. CyberShake assesses earthquake hazards for the Southern California Earthquake Center. Epigenomics, a collaboration between the USC Epigenome Center and the Pegasus Team, automates processes in genome sequence handling. These workflows, with a history of practical use, are ideal for evaluating optimization algorithm performance.

3.2. Simulation environment

The FogWorkflowSim simulator runs on the Eclipse Java IDE. Simulations are conducted on a 64-bit Windows 10 system with an Intel Core i7-M 640 @ 2.80 GHz and 6 GB of RAM. Each algorithm uses a population size of 50. In the GA, the crossover rate is set at 0.8 and the mutation rate at 0.1. For each workflow, simulations are performed 10 times to derive average performance. The experimental setup involves ten end devices, six fog virtual machines (VMs), and three cloud VMs. Server characteristics for the three IoT-cloud-fog layers and specific parameter configurations are detailed in Table 1.

Table 1. The IoT-fog-cloud environment parameter setting

Parameters	End device	Fog VM	Cloud VM
Processing rate (MIPS)	1000	1300	1600
Task execution cost ($)	0	0.48	0.96
Communication cost ($)	0	0.01	0.02
Working power (MW)	700	800	1600
Idle power (MW)	30	40	1300
Uplink bandwidth (Mbps)	20	10	1
Downlink bandwidth (Mbps)	40	10	10

3.3. Simulation results

The proposed algorithm’s performance is assessed in comparison to a simple method in [25]. Which is an offloading strategy implemented within the fog workflow simulator, using GA in [26] for the scheduling process. This comparison is made due to the effectiveness of this method in scenarios where specific segments of a task can be executed concurrently or when the available local resources are insufficient.

Figures 2 to 4 present the outcomes concerning makespan, cost, and energy consumption for the Montage. As anticipated, the metrics exhibit an increase as the number of tasks increases across scenarios involving 20, 60, and 100 tasks. Contrastingly, the LCO strategy outperforms in makespan reduction by 35.17%, 27.74%, and 26.29% compared to the simple method for 20, 60, and 100 tasks. The EO strategy closely follows, with reductions of 18.42%, 26.32%, and 29.26% for the same task counts. As tasks increase to 100, EO achieves top performance in makespan, attributed to extra waiting time in the IoT tier for LCO. Conversely, EBO and simple exhibit similar performance levels, with EBO increasing makespan by 18.1% for 20 tasks and decreasing it by 0.04%, 1% for 60, and 100 tasks, respectively, compared to the simple method.

The LCO strategy outperforms in cost reduction, decreasing total cost by 98.14%, 98.04%, and 97.28% compared to the simple method for 20, 60, and 100 tasks. EO follows closely with decreases of 32.71%, 51.11%, and 46.13%, while EBO shows results close to the simple method, decreasing cost by 1.97% at 20 tasks but causing 0.72% and 0.34% increases for 60 and 100 tasks, respectively, compared to the simple strategy. In energy use, LCO's higher local execution results in a 19.99%, 40.7%, and 46.96% increase compared to the simple method for 20, 60, and 100 tasks. EO, with a balanced job distribution strategy, increases energy consumption by 5.28%, 23.77%, and 25.22% for 20, 60, and 100 tasks. EBO closely aligns with the simple method, decreasing energy consumption by 0.32%, 0.02%, and 0.43% for 20, 60, and 100 tasks.

Figures 5, 6, and 7 display makespan, cost, and energy consumption outcomes for the CyberShake workflow. Latency-centric offloading (LCO) and efficient offloading (EO) outperform in makespan and cost across all tasks. LCO reduces makespan by 60.77%, 48.11%, and 31.48% for 30, 50, and 100 tasks, followed by EO with decreases of 60.58%, 48.11%, and 31.49%. EBO and the simple method exhibit relatively constant makespan performance for all tasks.

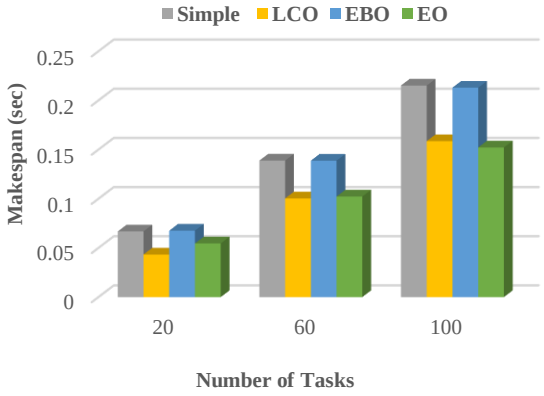

Figure 2. Montage makespan results

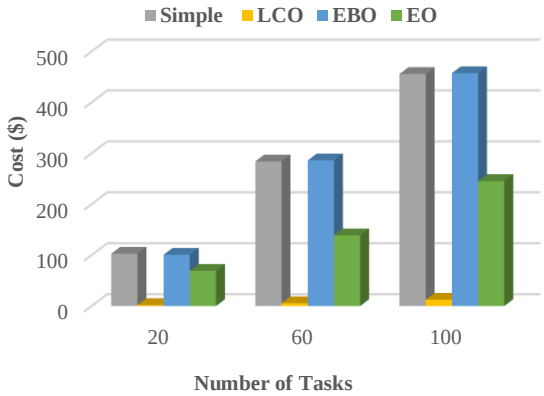

Figure 3. Montage cost results

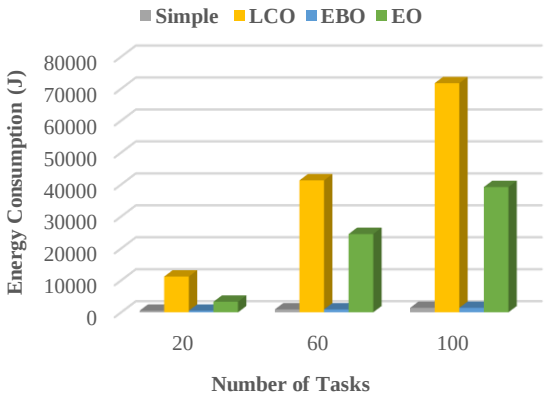

Figure 4. Montage energy consumption results

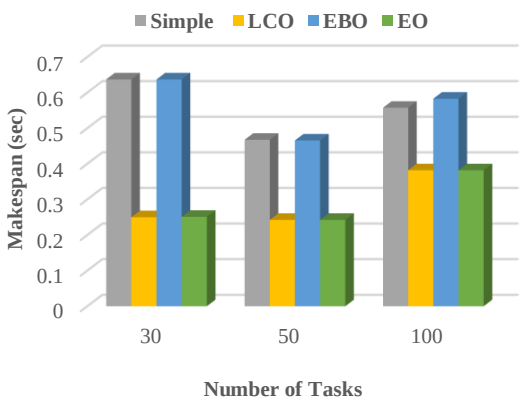

Figure 5. CyberShake makespan results

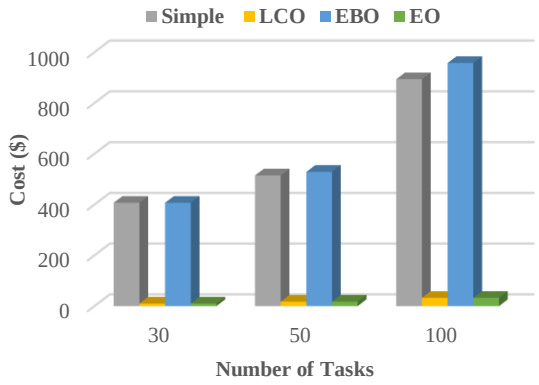

Figure 6. CyberShake cost results

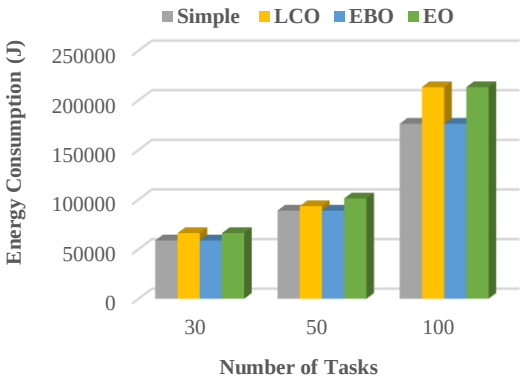

Figure 7. CyberShake energy consumption results

Regarding the cost metric, LCO and EO show better performance compared to other strategies, with LCO decreasing cost by 97.75%, 96.60%, and 96.38%, and EO decreasing cost by 97.51%, 96.59%, and 96.38% compared to the simple method for 30, 50, and 100 tasks, respectively. EBO and the simple method give relatively close results for the cost metric at 30 tasks, with EBO increasing cost by 2.77% compared to the simple method and by 7.17% for 100 tasks. In terms of energy consumption, EBO outperforms other proposed strategies. EBO and the simple method show relatively constant energy consumption for 30 tasks, with a 0.01% decrease for 50 tasks and a 0.04% increase for 100 tasks compared to the simple method. EO and LCO follow for 30 and 100 tasks, with EO increasing energy consumption by 12.40%, 20.91%, and LCO by 12.43%, 20.91% compared to the simple method. For 50 tasks, LCO increases energy consumption by 5.1% compared to the simple method, and EO by 13.74%.

Figures 8 to 10 illustrate makespan, cost, and energy consumption for the Epigenomics workflow. Metrics for 24 and 47 tasks are low but significantly rise with 100 tasks: makespan from under 10 seconds to over 50 seconds, cost from under $10,000 to over $50,000, and energy consumption from under 50,000 J to beyond 250,000 J. This increase is attributed to computationally demanding tasks, particularly aligning sequences with the reference genome [28], resulting in extended runtimes as the task count rises.

In terms of makespan and cost, LCO maintains superior performance, reducing makespan by 21.30%, 20.33%, and 23.86% and decreasing cost by 100%, 84.82%, and 86.20% compared to the simple method for 24, 47, and 100 tasks. A 100% cost decrease for 24 tasks implies all jobs are executed locally on end devices, indicating highly efficient task allocation for cost optimization. However, LCO exhibits poorer energy consumption performance compared to other strategies. The LCO algorithm assigns more tasks to end devices as the task count rises, resulting in a substantial cost reduction due to reduced expenses for data transmission and computation. This reliance on end devices leads to increased energy consumption: LCO increases energy consumption by 44.46%, 63.60%, and 113.66% compared to the simple method for 24, 47, and 100 tasks, respectively. EBO secures the second position, achieving a negligible 1.33% reduction compared to the simple method for 24 tasks, with similar, almost negligible results for 47 and 100 tasks.

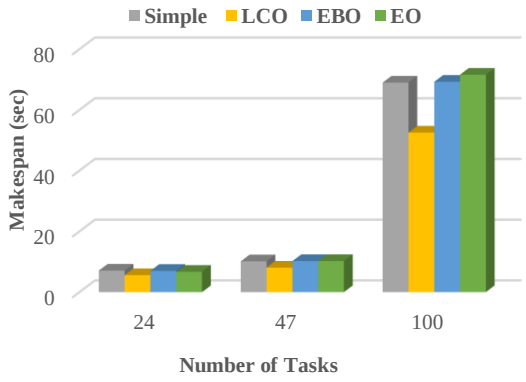

Figure 8. Epigenomics makespan results

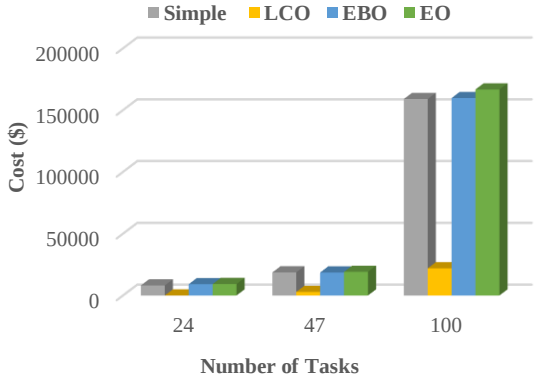

Figure 9. Epigenomics cost results

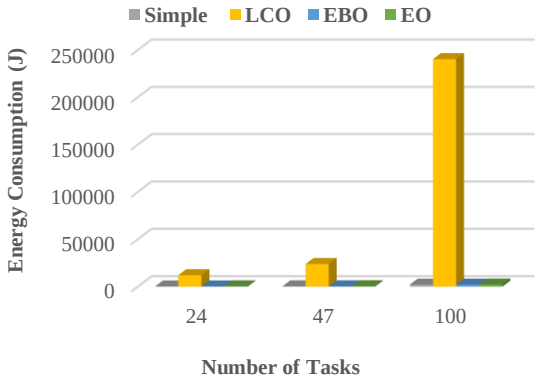

Figure 10. Epigenomics energy consumption results

Ultimately, EO performs between LCO and EBO. For 24 tasks, EO reduces makespan by 5.55% compared to the simple method but increases it by 0.93% and 3.77% for 47 and 100 tasks, respectively. However, EO consistently increases cost compared to the simple method, with changes of 13.94%, 2.44%, and 4.9% for 24, 47, and 100 tasks. It also results in higher energy consumption, with increases of 28.18%, 29.83%, and 5.60% compared to the simple method for 24, 47, and 100 tasks, respectively.

4. CONCLUSION

This study proposes three strategies: LCO, EBO, and EO as offloading strategies in IoT-fog-cloud environments, with a focus on real-time applications. LCO is best for time-sensitive tasks but can be more energy-intensive. EBO focuses on energy efficiency, which is beneficial for long-term and computationally demanding tasks. EO seeks a balanced trade-off between time and energy, suited for using resources more effectively. This work employs genetic algorithms for task scheduling because they excel at handling complex solution spaces and dynamic situations. Combining offloading techniques with task scheduling algorithms provides efficient task execution. Overall, these strategies contribute to the optimization and reliability of IoT-fog-cloud systems in different environments, offering specified solutions for different task requirements and environmental constraints. Regarding future work, different types of workflows will be incorporated, such as LIGO and SIPHT, and the number of tasks in each workflow will increase.

REFERENCES

[1] E. Deelman *et al.*, "The future of scientific workflows," *The International Journal of High Performance Computing Applications*, vol. 32, no. 1, pp. 159–175, Jan. 2018, doi: 10.1177/1094342017704893.
[2] Z. Ahmad *et al.*, "Scientific workflows management and scheduling in cloud computing: Taxonomy, prospects, and challenges," *IEEE Access*, vol. 9, pp. 53491–53508, 2021, doi: 10.1109/ACCESS.2021.3070785.
[3] M. Sardaraz and M. Tahir, "A hybrid algorithm for scheduling scientific workflows in cloud computing," *IEEE Access*, vol. 7, pp. 186137–186146, 2019, doi: 10.1109/ACCESS.2019.2961106.
[4] B. Ludäscher, M. Weske, T. McPhillips, and S. Bowers, "Scientific workflows: business as usual?" in *Business Process Management: 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings 7*, 2009, pp. 31–47.
[5] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," *Journal of Grid Computing*, vol. 3, no. 3–4, pp. 171–200, Sep. 2005, doi: 10.1007/s10723-005-9010-8.
[6] S. Jha, S. Lathrop, J. Nabrzyski, and L. Ramakrishnan, "Incorporating scientific workflows in computing research processes," *Computing in Science & Engineering*, vol. 21, no. 4, pp. 4–6, Jul. 2019, doi: 10.1109/MCSE.2019.2917987.
[7] M. A. Miller, W. Pfeiffer, and T. Schwartz, "The CIPRES science gateway," in *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, Jul. 2011, pp. 1–8, doi: 10.1145/2016741.2016785.
[8] Rubing Duan, R. Prodan, and Xiaorong Li, "Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 29–42, Jan. 2014, doi: 10.1109/TCC.2014.2303077.
[9] Y. Zhao *et al.*, "A service framework for scientific workflow management in the cloud," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 930–944, Nov. 2015, doi: 10.1109/TSC.2014.2341235.
[10] N. P. Sodinapalli, S. Kulkarni, N. A. Sharief, and P. Venkatareddy, "An efficient resource utilization technique for scheduling scientific workload in cloud computing environment," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 1, pp. 367–378, Mar. 2022, doi: 10.11591/ijai.v11.i1.pp367-378.
[11] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus toolkit for market-oriented cloud computing," in *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*, 2009, pp. 24–44.
[12] J. Geetha, S. Zuveria Kottur, R. Ganiga, D. S. Jayalakshmi, and T. Surabhi, "Analysis of rank-based latency aware fog scheduling using validating internet of things at large scales," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 26, no. 3, pp. 1502–1511, Jun. 2022, doi: 10.11591/ijeecs.v26.i3.pp1502-1511.

[13] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: issues and challenges," *IEEE Network*, vol. 30, no. 4, pp. 46–53, Jul. 2016, doi: 10.1109/MNET.2016.7513863.
[14] T. Q. Quek, M. Peng, and O. (Eds. Simeone), *Cloud radio access networks: Principles, technologies, and applications*. Cambridge University Press, 2017.
[15] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, Apr. 2018, doi: 10.1109/JIOT.2017.2788802.
[16] A. H. Shamman, H. A. Alasadi, H. A. Ameen, Z. I. Rasol, and H. M. Gheni, "Cost-effective resource and task scheduling in fog nodes," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 27, no. 1, pp. 466–477, Jul. 2022, doi: 10.11591/ijeecs.v27.i1.pp466-477.
[17] K. J. Naik, "A cloud-fog computing system for classification and scheduling the information-centric IoT applications," *International Journal of Communication Networks and Distributed Systems*, vol. 27, no. 4, 2021, doi: 10.1504/IJCNDS.2021.119208.
[18] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," *Parallel Computing*, vol. 62, pp. 1–19, Feb. 2017, doi: 10.1016/j.parco.2017.01.002.
[19] Y. kothyari and A. Singh, "A multi-objective workflow scheduling algorithm for cloud environment," in *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Feb. 2018, pp. 1–6, doi: 10.1109/IoT-SIU.2018.8519931.
[20] A. M. Manasrah and H. Ba Ali, "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–16, 2018, doi: 10.1155/2018/1934784.
[21] M. Farid, R. Latip, M. Hussin, and N. A. W. Abdul Hamid, "A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing," *Symmetry*, vol. 12, no. 4, Apr. 2020, doi: 10.3390/sym12040551.
[22] T. S. Nikoui, A. Balador, A. M. Rahmani, and Z. Bakhshi, "Cost-aware task scheduling in fog-cloud environment," in *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, Jun. 2020, pp. 1–8, doi: 10.1109/RTEST49666.2020.9140118.
[23] P. M. Rekha and M. Dakshayini, "Efficient task allocation approach using genetic algorithm for cloud environment," *Cluster Computing*, vol. 22, no. 4, pp. 1241–1251, Dec. 2019, doi: 10.1007/s10586-019-02909-1.
[24] D. Subramoney and C. N. Nyirenda, "A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec. 2020, pp. 760–767, doi: 10.1109/SSCI47803.2020.9308549.
[25] X. Liu *et al.*, "FogWorkflowSim: an automated simulation toolkit for workflow performance evaluation in fog computing," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov. 2019, pp. 1114–1117, doi: 10.1109/ASE.2019.00115.
[26] D. Subramoney and C. N. Nyirenda, "Multi-swarm PSO algorithm for workflow scheduling in cloud-fog environments," *IEEE Access*, vol. 10, pp. 117199–117214, 2022, doi: 10.1109/ACCESS.2022.3220239.
[27] X. Zhao, L. Zhao, and K. Liang, "An energy consumption oriented offloading algorithm for fog computing," in *Quality, Reliability, Security and Robustness in Heterogeneous Networks: 12th International Conference, QShine 2016, Seoul, Korea, July 7–8, 2016, Proceedings 12*, 2017, pp. 293–301.
[28] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *2008 Third Workshop on Workflows in Support of Large-Scale Science*, Nov. 2008, pp. 1–10, doi: 10.1109/WORKS.2008.4723958.

BIOGRAPHIES OF AUTHORS

Marwa Gamal was born in London in 1985. She received the B.Sc. and M.Sc. degrees in computer and control engineering from Suez Canal University, Egypt, in 2007 and 2012, respectively, and the Ph.D. degree in computer engineering from Suez Canal University, Egypt, in 2019. She is currently an assistant professor with the Electrical Engineering Department, Computer and Control branch, Suez Canal University, Ismailia, Egypt. Her research interests include the area of IoT, fog computing, cloud computing, deep learning, and artificial intelligence. She can be contacted at email: marwa_gamal@eng.suez.edu.eg.

Samar Awad was born in Portsaid, Egypt in 1986. She received the B.S. degree in computers and control engineering from Suez Canal University, Portsaid, Egypt, in 2008, the M.Sc. degree in computers engineering from Suez Canal University, Egypt, in 2021. She is currently an assistant lecturer within the Electrical Engineering Department, Computers and control branch, Suez Canal University, Ismailia, Egypt. Her research interests include wireless sensor networks, internet of things (IoT), and fog computing. She can be contacted at email: Samar_Awad@eng.suez.edu.eg.

Rehab F. Abdel-Kader is a professor of computer engineering within the Faculty of Engineering, Port Said University. She earned her Ph.D. degree in computer engineering from Auburn University, Alabama, USA, in 2003. Following her doctoral studies, she contributed as an assistant professor in the Electrical Engineering Department at Georgia Southern University, GA, USA, where she honed her expertise for a span of two years before returning to her home country, Egypt. She serves as the vice dean of Graduate Studies and Research within the Faculty of Engineering at Port Said University. Her scholarly pursuits encompass a strong focus on cutting-edge fields, with a particular emphasis on artificial intelligence and computer vision. She can be contacted at email: rehabfarouk@eng.psu.edu.eg.

Khaled Abd El Salam was born in Egypt in 1973. He received the B.S. and M.Sc. degree in computer engineering from Suez Canal University in 2003 and the Ph.D. degree in computer engineering from Suez Canal University in 2010. From 2000 to 2013, he was a computer instructor in The Higher Institute for Hotels and Tourism Management "HIHTM" Lucerne, Hurghada, Egypt. Then from 2013 to 2018, he was an associate professor in Faculty of Engineering, Sinai University, Egypt. Since 2018, he has been associate professor in Faculty of Engineering, Suez Canal University, Ismailia, Egypt. He can be contacted at email: Khaled.abdelsalam@eng.suez.edu.eg.