Clustering-based Resource Management for Consumer Cost Optimization in IoT Edge Computing Environments

Mohsen Denden, Mahdi Jemmali, Wadii Boulila *Senior Member, IEEE*, Mukesh Soni, Faheem Khan, and Jawad Ahmad *Senior Member, IEEE*

***Abstract*—Edge computing emerges as a pivotal model in the era of next-generation consumer electronics and the emerging challenges of multimodal data-driven decision-making. Specifically, edge computing offers an open computing architecture for the vast and diverse consumer multimodal data generated by mobile computing and Internet of Things (IoT) technologies. While edge computing is instrumental in optimizing latency and bandwidth control in processing consumer multimodal data, the viability of employing edge resources is complicated by high service costs and the complexities of managing multimodal data diversity. This study introduces an innovative optimization method for distributing multimodal data on edge storage, considering both the I/O (input/output) speed and the overall distribution cost. The core part of our approach is the deployment of intelligent algorithms that ensure equitable data distribution across storage servers, thus eliminating unused space and reducing extra costs. Given the complexity of this NP-hard (non-deterministic polynomial-time) challenge, our study reveals a unique model incorporating an edge-broker component in combination with novel algorithms. The proposed algorithms aim to harmonize data distribution and reduce resource allocation expenses in a multimodal edge environment. Our proposed approach achieves excellent results, highlighting the efficacy of the proposed algorithms in several parameters such as makespan, cost, multimodal data security, and total processing time. Empirical tests reveal that the BCA (Best Clustering Algorithm) performs best, achieving a minimum load balancing rate of 92.2%, an average variance of 0.04, and an average run time of 0.56 seconds.**

***Index Terms*—Consumer Cost, edge workflow, clustering, randomization, load balancing.**

Corresponding authors: Wadii Boulila (e-mail: wboulila@psu.edu.sa); Mahdi Jemmali (e-mail: mjemmali@sharjah.ac.ae)

M. Denden, Department of Computer and Information Technologies, College of Telecommunication and Information, Riyadh CTI, Technical and Vocational Training Corporation $TVTC$ Saudi Arabia; Department of Computer Science, Higher Institute of Applied Sciences of Sousse, University of Sousse, Sousse, 4000, Tunisia (e-mail: mohsen@cti.edu.sa)

M. Jemmali, Department of Computer Science, College of Computing and Informatics, University of Sharjah, Sharjah, United Arab Emirates, with the MARS Laboratory, University of Sousse, Sousse, Tunisia, and with the Department of Computer Science, Higher Institute of Computer Science and Mathematics, University of Monastir, Monastir, 5000, Tunisia

W. Boulila, Robotics and Internet-of-Things Laboratory, Prince Sultan University, Riyadh, Saudi Arabia, and with RIADI Laboratory, National School of Computer Science, University of Manouba, Manouba, Tunisia

M. Soni is with the Department of CSE, University Centre for Research & Development Chandigarh University, Mohali, Punjab-140413, India

F. Khan is with the Department of Computer Engineering, Gachon University, Seongnam-si, South Korea (e-mail: faheem@gachon.ac.kr)

J. Ahmad is with the School of Computing, Engineering and the Built Environment, Edinburgh Napier University, United Kingdom (e-mail: J.Ahmad@napier.ac.uk)

I. Introduction

Edge computing provides services for applications with strict quality of service constraints, including low-latency 5G and IoT applications. However, due to their limited computing capacity, edge data centers are not able to respond to a large number of user queries. Additionally, the diversity of user data creates significant challenges in analysis speed, processing, and intelligent decision-making. Therefore, optimizing resource management is crucial to delivering scalable services to end-users. Several scheduling applications can be considered for intelligent decision-making [1], [2], [3], [4], [5], [6]. Traditional optimization methods are no longer suitable, and there is a growing demand for the development of faster and more advanced algorithms.

Many researchers have studied the problem of planning and resource allocation in an edge environment. Some of them were able to provide solid formulations and approaches and developed heuristic classification and algorithms suitable for this problem [7], [8], [9], [10], [11]. In [12], the authors summarized the different scheduling algorithms by providing a thorough classification and a comparative study of the main algorithms developed in recent years. The interest was mainly focused on scheduling issues in the edge environment. The categorization of models, algorithms, and heuristics was organized according to the design basis. The advantages and disadvantages of each algorithm have been discussed in more detail. Additionally, reference [12] summarized future directions related to scheduling issues. The authors in [13] discussed the basic task scheduling concept in a cloud environment by presenting a literature review of the scheduling issues. Various meta, evolutionary, physical, and hybrid planning techniques were reviewed, depending on the nature of the planning problem. The objective was to classify central resource allocation systems [13].

The application of scheduling algorithms in edge computing environments has gained increasing interest from the scheduling and optimization community [14],[15]. The efficient distribution of workflow tasks across available resources necessitates complex algorithms due to the complex nature of parameters such as cost, delay, and Quality of Service (QoS). [16], [17]. In [18], the authors criticize existing scheduling approaches, arguing that the distribution of tasks is not founded on robust approaches and assumptions. It has been highlighted that most developed algorithms are only effective

under specific conditions. The authors developed and tested several algorithms for application in both homogeneous and heterogeneous environments. In [19], the authors employed a correlation spectral-density technique to develop heuristic scheduling tasks for hybrid storage. In [20], the authors combined genetic algorithms with other heuristics to develop novel hybrid heuristics, termed $HSGAs$, for effectively ordering workflow graphs. Furthermore, the authors optimized makespan and load balancing using the same approach. Recently, the authors in [21] proposed novel scheduling algorithms for drone battery management. These algorithms can be leveraged across various application domains. Controlling and reducing costs of edge services are important factors for many businesses. If proper measures are not taken carefully, bills will increase unnecessarily. The optimization and minimization of this cost have been the subject of several bibliographic researches.

Different models are presented in the literature to optimize costs in data storage [22]. In [23], the authors made a new database for looking at the effects of only keeping some analysis requests from the Database Management System (DBMS) to the Amazon Web Services (AWS) service. They demonstrated that migrating some DBMS components to the AWS platform could be more cost-effective. In [24], the authors addressed the problem of big data stream planning. They proposed a simulation model to manage file transfers, taking into account cost and time constraints. In [25], the authors developed a cost optimization model for data in the cloud environment. Scientific workflow is modeled as a directed graph Directed Acyclic Graphs (DAG). In this work, general-purpose cloud benchmarks were used to evaluate their model and demonstrated that their model could minimize the cost of workflow running under time constraints. The authors of [26] proposed a workflow scheduling model to optimize cost in the case of scientific workflow applications. The developed model was utilized to optimize the makespan and overall computational costs.

In the article [27], the authors considered that applications used in cloud-based industrial control systems are more susceptible to threats. Security for these types of applications should be highly prioritized. A safety-aware dynamic scheduling method based on the Particle Swarm Optimisation (PSO) algorithm was developed in [27]. It has been demonstrated that via scheduling policy [27], one can achieve a balance between safety and scheduling performance. The article [28] proposed a QoS model for evaluating performance in data centers. In this work, the cross-entropy-based algorithm was used to assess QoS. Previous literature reveals that several researchers have studied the development of IoT technology and contributed to developing many models and methods to make this technology more efficient and cost-effective. The combination of IoT and AI technologies, also known as AIoT, is in high demand. Researchers are attempting to leverage the strengths of each technology. The paper [29] presents the design of the basic technologies of AIOT. The authors in [30] proposed an algorithm to assess network health, reliability, and efficiency for an edge application. This work aims to minimize the network delay and evaluate the edge network against the cloud network. The work discussed in [31] proposed collaborative scheduling methods based on the analysis and synthesis of calculation tasks in edge computing. In [32], a study of edge server network design is proposed. They aimed to balance the cost of network building and the density in the edge computing environment. A model of cost-density constraint as a constrained optimization problem is proposed, and various tests were conducted to solve this problem at large and small scales. The techniques used in [33], [34], [35], [36], [37], [38] and [39] can be enhanced by clustering algorithms.

The data scheduling problem in edge environments has been extensively studied, as evidenced by the related literature. The main aim is to find a solution that ensures a proper allocation of edge resources to minimize costs. However, cost optimization is a complex issue and needs further investigation. The developed models, applied under specific requirements, need further refinement to support workflows and reduce overall costs. In this work, we have developed a heuristic scheduling method for minimizing edge storage costs.

Service providers offer great potential to reduce the cost of data storage, but the overall cost remains high. Using intelligent scheduling algorithms helps reduce this storage cost by redirecting data to the right places, minimizing unused space, and avoiding unnecessary data transfer between resources. Our contribution consists of developing and testing new algorithms to reduce the storage gap between the different available resources and thus avoid additional costs.

The main aim of this work is to reduce global storage consumer costs by optimizing storage resource allocation. To achieve this, we will develop intelligent scheduling strategies and new algorithms. This approach prevents unbalanced loads, reduces the need for future data transfers, minimizes unnecessary migration, and maintains high security. Several algorithms are proposed, and tests are analyzed to achieve the aforementioned. A few algorithms are based on the clustering method, such as the clustering two-groups algorithm, and others are based on the randomization method, such as the random-clustering two-groups algorithm. The primary objective of developing these six algorithms is to identify the most efficient data scheduling method, minimizing resource capacities and processing time differences.

The remainder of this paper is structured as follows: Section 2 provides a detailed problem description. Section 3 discusses the proposed model for load balancing in edge computing. Section 4 explains the proposed load-balancing algorithms. Section 5 presents experimental results and discussions. The conclusion is presented in Section 6.

II. Problem description

The accelerated development of IoT and AIoT has sparked fundamental interest in the volume of data that needs to be supported. However, considering the limited resource capacity and financial budgets of most consumers, applying smarter models can help mitigate this challenge.The following paragraph highlights the problem addressed in this work.

Fig 1 shows the IoT architecture tree [40]. IoT connects objects and devices equipped with sensors, enabling them to

This article has been accepted for publication in IEEE Transactions on Consumer Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCE.2024.3414929

exchange data among themselves and with other systems. IoT applications cover various domains, each generating a wide variety of data.

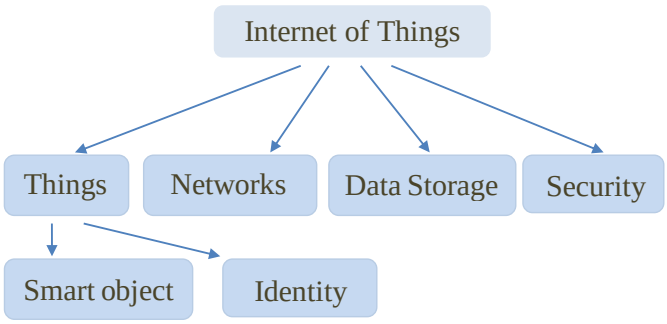

Fig. 1. IoT architecture tree

Edge computing technology has several applications, such as autonomous vehicles, healthcare, security solutions, agriculture, and industry. Utilizing edge computing (Figure 1) for processing and data storage can enhance service speed and stability. However, traditional data flow management methods can result in server overloads and network service denials. The overall cost of resources can increase significantly. A smart, balanced data storage planning approach can reduce delays and minimize the overall cost. Therefore, developing collaborative and selective scheduling models is important to meet the demands of real-time IoT applications. The significant increase in data volumes (Figure 2), primarily from IoT networks, has complicated analysis in traditional data centers. Edge computing aids in minimizing the number of processes required in the cloud. A primary advantage of edge computing is the reduced data transmission times and the lowered hardware and storage process costs. The following paragraph describes the cost formulation and mathematical model used in this study.

Generally, to estimate the overall cost of storage operations, one should begin by calculating fixed and recurring server expenses. Fixed costs include the expenses for server purchase licenses, operating systems, and, notably, memory and storage space. Concurrently, recurring costs encompass network connectivity, maintenance, administration, and network monitoring. The literature reveals that various heuristics and meta-heuristics have been devised to optimize storage costs. Despite these advancements, the wastage of storage resources persists and remains a significant challenge. Researchers have previously outlined that a vacuum of reserved capacity and space exists, which can lower storage costs and CPU cycles.

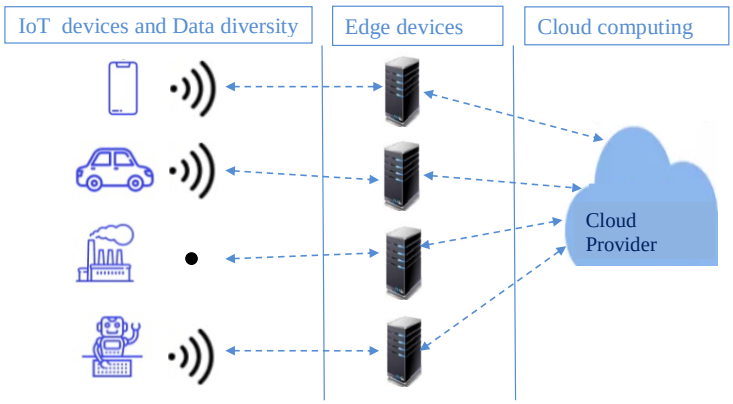

Fig. 2. IoT and Edge devices

The problem under study involves developing a method that reduces the overall storage cost in edge computing while ensuring an equitable distribution of files.

The description of variables is provided in Table I.

TABLE I
VARIABLE NOMENCLATURES

Variable	Description
S_F	Set of files
S_S	Set of servers
f_n	Number of files
S_n	Number of servers
j	File index
i	server index
K	Number of tasks
S_i	Server i
C_u	Cost unit
C_tot	Total Cost
F_j	File j
fs_j	The size of the file j
us_j	The used space when the file j is stored
Ts_i	The total space in S_i when finishing the storing
Ts_{min}	The minimum used space for all S_i, $i \in \{1, \cdots, S_n\}$

As outlined in [25], one can estimate the total cost in edge computing through Equation (1)):

$$C_{tot} = \sum_{i \in S_n, k \in K} (P_i.H_{i,k} + P^R + C^T_{i,s}).T_{i,k} \quad (1)$$

- $T_{i,k}$: integer, it represents the number of tasks that are proposed on server i type k.
- $H_{i,k}$: Running time of servers i and type k.
- P_i: Fee (in US dollars) for running a server of type k for one hour.
- $C_{k,s}$: Cost of data transfer between servers for type k and a storage site S.
- P^R: Price per task for queuing service.

In the rest of this work, we will consider k and s as constants. Our goal is to minimize the term $(P_i.H_{i,k})$ in Equation (1) with :

$$P_i = (Ts_{min} + Gv_i/s_n) * C_u. \quad (2)$$

and

$$Gvi = (Ts_i - Ts_{min}). \quad (3)$$

Gvi : is the gap space between servers i and the lowest server space.
The total gap between server-used spaces is denoted by Gv and given in Equation (4).

$$Gv = \sum_{i=1}^{S_n} (Ts_i - Ts_{min}). \quad (4)$$

Minimizing the term $(P_i.H_{i,k})$ as considered in Equation (1) is equivalent to reducing the term Gv. The remainder of this paper will discuss algorithms that aim to achieve this goal.

The following example illustrates the key function of the scheduling problem.

Example 1: Let $f_n = 7$ and $s_n = 2$. Table II represents the size fs_j for each file F_j.

TABLE II
SIZE-FILE VALUES FOR EXAMPLE 1.

F_j	1	2	3	4	5	6	7
fs_j	20	10	13	5	12	8	15

Let us assume that we have the schedule shown in Fig 3. This schedule is based on the smallest first algorithm. It is observed that files $\{1, 2, 3, 4\}$ are assigned to server 1, while files $\{5, 6, 7\}$ are assigned to server 2.

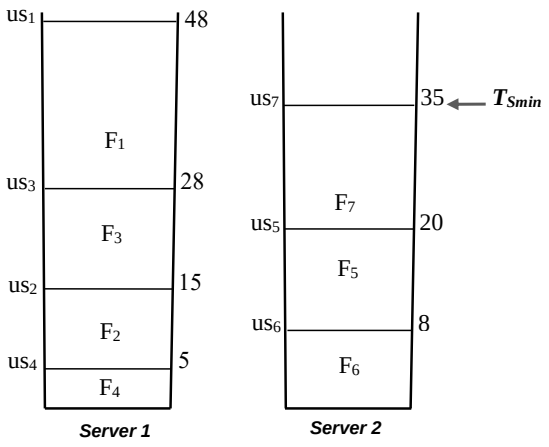

Fig. 3. Files schedule for Example 1.

The total used space in server 1 is 48, as shown in Fig 3. However, the total used space in server 2 is 35. Thus, the used space 35 is the minimum used space, denoted by $Ts_{min} = 35$. The gap between the used spaces is denoted by Gv, which is calculated as $Gv = Ts_1 - Ts_{min} = 48 - 35 = 13$.

This work aims to minimize the Gv gap across servers. An algorithm that achieves a minimum gap of less than 13 is necessary.

Minimizing the Gv is the primary goal of this work. Adopting a different schedule than the one presented in Figure 3, example 2 demonstrates improved results and a minimum gap of less than 13.

Example 2: The same instance detailed in Table III is taken in this example. The schedule illustrated in Fig 4 is based on the biggest first algorithm. This schedule shows that the total used space in server 1 is 40. However, the total used space in server 2 is 43 ($Ts_{min} = 40$). The gap between the used spaces is $Gv = Ts_2 - Ts_{min} = 43 - 40 = 3$.

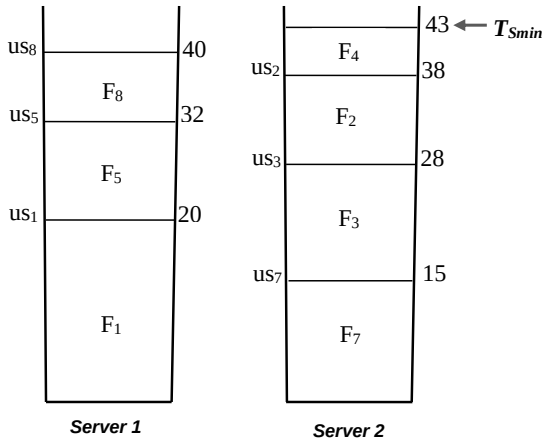

Fig. 4. Files schedule for Example 2.

The schedule illustrated in example 2 gives a better result than the schedule in example 1. The gained space here is 10 units. The outcome of the scheduling instance depicted in example 2 can serve as baseline information for predicting and estimating capacity allocation. This approach dynamically determines capacity needs and subsequently reduces allocation costs.

III. PROPOSED MODEL FOR LOAD BALANCING

The research discussed above underlines the challenges of identifying optimal scheduling solutions in an edge computing environment. Many researchers have outlined multiple research directions that warrant further exploration and deepening.

Within a computing environment, models developed by [41] and optimization methods discussed in the literature often provided optimal solutions. Due to their complexity, these solutions may require significant computational time. The primary issue involves dispatching files to each storage resource to minimize allocation costs. The architecture shown in Figure 5 consists of the following model components:

- File manager is a mapping event in which files to be stored should be grouped into a batch before being sent.
- The edge broker is an intermediary between AIoT customers and edge service providers.
- Developed heuristics aim to deliver appropriate scheduling solutions, ensuring the shortest possible makespan. These heuristics consider the incoming workflow, queued data, and resource allocation state.
- Edge contains all servers capable of directly receiving storage data.

- Cloud resources comprise all available Virtual Machines (VMs) that can accommodate additional storage data

The main idea of this study is to assign data to suitable servers in the Edge environment. Intelligent scheduling of real-time data flow is an essential process for guiding files to be stored. After receiving user requests, the task manager component should gather accepted files, analyze them according to customer constraints, and estimate the needed capacities.

The resource manager sweeps up all available resources and collects information about Edge servers. This component translates this information to the task scheduler. The load balancer component instantly calculates the percentage of used capacities in each server. This information is received by the resource manager component. The task manager gathers information again, checks the developed heuristic results, and assigns each file to the suitable server. The developed heuristics component, the heart of our work, operates in collaboration with the task scheduler. It collects the necessary information and dynamically calculates the best solution to assign and dispatch each file to its corresponding servers. Fig 5 shows the proposed model.

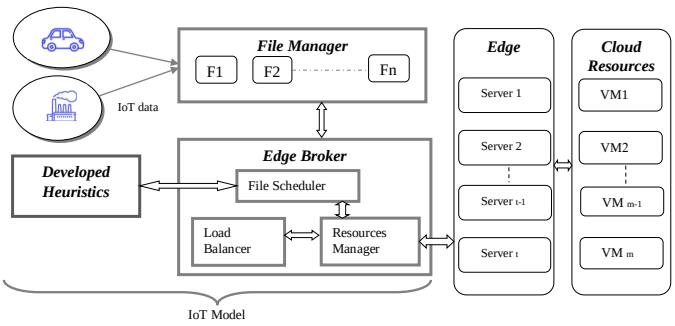

Fig. 5. Novel proposed model for load balancing in Edge computing

The main idea of this study is to assign IoT data to the appropriate Edge storage servers. Dynamic scheduling is important in guiding tasks within the Edge environment. Without a scheduling process, global costs can increase, and resource capacity can be lost. To apply the scheduling process, we need to analyze the storage resources' number and state.

In this work, we focus on developing scheduling algorithms that play the primary role of the "Load balancer" component. The following section details the proposed algorithms.

IV. Load Balancing Algorithms

Six algorithms for load balancing in the Edge are presented. The first one is based on the dispatching rules. The second one is based on the clustering method, which uses two groups to classify files to be scheduled. The third algorithm is similar to the previous one, but we add the probabilistic aspect to choose between groups. The excluding-files clustering two-groups algorithm will be presented as the fourth algorithm. The randomization method is applied to propose the random-clustering two-group algorithm as the fifth algorithm. Finally, the last algorithm is based on the choice of two proposed algorithms, and the best solution will be picked.

A. Longest File First algorithm (LFF)

The initial stage in this technique is to arrange all files in decreasing order of size. Then, we assign the first file to the server with the minimum total used space until all files are scheduled. This rule-based dispatching algorithm is characterized by its fast time execution. This algorithm is denoted by LFF. The complexity of LFF is $O(nlogn)$. Hereafter, $SCHD(F)$ is the procedure that schedules the file F on the server with the minimum used space. We denote by $DecR()$ the procedure that sorts the set of given files according to the decreasing order of their size.

Algorithm 1 Longest File First algorithm (LFF)

1: Call $DecR(S_F)$
2: Call $SCHD(F)$
3: Calculate Gv
4: Return Gv

B. Standard Clustering Two-groups algorithm (SCT)

This approach is mostly based on the clustering method. This clustering detects two file clusters, indicated by the symbols $G1$ and $G2$. The initial stage of this approach is sorting the files in decreasing order of size. The second step of this algorithm is constructing $G1$ and $G2$. At this stage, $G1$ and $G2$ are empty. The final step is distributing the files and starting $G1$ and $G2$. The first file is assigned to $G1$ and the second to $G2$. The third file is assigned to the group with the minimum used space until all files are assigned. Now, $G1$ and $G2$ are constructed. The fourth step involves assigning the files to servers. This phase is based on a random selection of $G1$ and $G2$, with the probability $sigma$ for selecting $G1$ and the probability $1 - sigma$ for selecting $G2$. Once the group is selected, the first file is picked and assigned to the server with the minimum total used space. This procedure will be repeated 2500 times.

We denoted by $FFG(1)$ and $FFG(2)$ the first file in the group $G1$ and the first file in the group $G2$, respectively.

Algorithm 2 Standard Clustering Two-groups algorithm (SCT)

1: Call $DecR(S_F)$
2: Construct $G1$ and $G2$
3: **for** ($ter = 1$ to 2500) **do**
4: **for** ($j = 1$ to f_n) **do**
5: dr =random(1,2)
6: **if** ($dr = 1$) **then**
7: Call $SCHD(FFG1)$
8: **else**
9: Call $SCHD(FFG2)$
10: **end if**
11: **end for**
12: Calculate Gv_{ter}
13: **end for**
14: Determine $Gv = \min_{1 \leq ter \leq 2000} Gv_{ter}$

C. Choosing Clustering Two-groups algorithm (CCT)

This algorithm uses the same method applied in Subsection IV-B. The difference is in the fourth step. Indeed, the choice between $G1$ and $G2$ can be either by applying the probability σ once or by choosing the group with the maximum used space. Once the group is selected, the first file is picked and assigned to the server with the minimum total used space. This procedure will be repeated 2500 times. We denoted by $MinG(X1, X2)$ the function that can return the group that has the maximum used space between two given groups as inputs $X1$ and $X2$.

Algorithm 3 Choosing Clustering Two-groups algorithm (CCT)

```
Call DecR(S_F)
Construct G1 and G2
for (ter = 1 to 2500) do
    for (j = 1 to f_n) do
        dr =random(1,2)
        if (dr = 1) then
            Call SCHD(FFG(1))
        else
            Call SCHD(FFG(2))
        end if
        if (j ≠ f_n) then
            Call G = MinG(G1, G2)
            Call SCHD(FFG(G))
        end if
    end for
    Calculate Gv_ter
end for
Determine Gv = min_{1≤ter≤2500} Gv_ter
```

D. Excluding-files Clustering Two-groups algorithm (ECT)

First, we sort the files in decreasing order of size. Next, we exclude the s_n longest files, denoting the set of these files as FL. The set of the remaining files is represented by FR. Finally, s_n is scheduled to exclude files on servers individually. This procedure is repeated 2,500 times. We denote $SCT()$ as the procedure detailed in Algorithm 2.

Algorithm 4 Excluding-files Clustering Two-groups algorithm (ECT)

```
Call SCT(FR)
SCHD(FL)
Calculate Gv
Return Gv
```

E. Random-Clustering Two-groups algorithm (RCT)

This algorithm is based on the randomization method. Firstly, we apply the same procedure to generate two groups like SCT. Once the group is chosen, we choose randomly between the first and the second file in the selected group. This randomization is based on probability α to select the first file and the probability $1 - \alpha$ to select the second file. This procedure will be repeated 3500 times, and the best solution will be picked.

F. Best Clustering algorithm (BCA)

BCA algorithm is constructed by calling the SCT and RCT, and the minimum value will be picked. The best result is chosen between SCT and RCT, and the final value of BCA is returned. This demonstrates that none of the proposed algorithms have dominance. Indeed, the experimental results confirm that in Table IV, the percentage of SCT is 75.7%, and the percentage of RCT is 51.8%. However, when we calculate the minimum between SCT and RCT, the percentage of this new value is 92.2%. The complexity of BCA is $O(n^2)$.

V. Experimental results and discussions

This section presents the experimental results of executing the proposed heuristics to determine the lower and upper bounds of fs_j. The experiments were carried out using Microsoft Visual C++. The uniform distribution is denoted by $U[]$, while the normal distribution is represented by $N[]$. This study considers five classes as follows:

- Class 1: $fs_j \in U[50, 200]$.
- Class 2: $fs_j \in U[150, 500]$.
- Class 3: $fs_j \in U[100, 500]$.
- Class 4: $fs_j \in N[300, 50]$.
- Class 5: $fs_j \in N[400, 100]$.

The selected total number of files f_n and the selected total number of servers S_n are shown in Table III.

TABLE III
Number of files and servers' distribution

f_n	S_n
10,30,50	3,4,5
100,300,500	10,20,30
1000,2000,3000	10,20,30,50

In this context, for each class and each f_n and S_n, 10 instances were generated. Table III shows the number of the generated instances $(3 \times 3 + 3 \times 3 + 3 \times 4) \times 10 \times 5 = 1500$.

The dataset used in this study is available for download from [42].

The metrics used in this work are presented as follows:

- A^* denotes the minimum Gv values retrieved after executing the six considered algorithms.
- A represents Gv values obtained by the presented algorithm.
- Prg is the percentage for each algorithm to reach A^*.
- $GP = \frac{A-A^*}{A}$, if A approaches to 0 then G is equal to 0. GP values are between A^* and A
- AG denotes the GP average for a fixed instance number.
- T is the running time in seconds, or "-" if the time is less than 0.001 s.

A. Global results

Table IV shows the presentation of the global results for all six considered algorithms according to Prg, AG, and T. Best results are achieved by BCA as depicted in Table IV, with $Prg = 92.2\%$, an average gap of 0.04, and a running duration of 0.56 seconds. SCT has achieved the second-best results with $Prg = 75.7\%$. Surprisingly, ECT gives bad results with only $Prg = 3.2\%$. Table IV shows no dominance between all proposed algorithms.

TABLE IV
GLOBAL RESULTS PRESENTATION FOR ALL ALGORITHMS

	LFF	SCT	CCT	ECT	RCT	BCA
Prg	22.3%	75.7%	30.3%	3.2%	51.8%	92.2%
AG	0.45	0.08	0.39	0.74	0.24	0.04
T	-	0.20	0.15	0.20	0.37	0.56

B. File number variation

Table V shows the results for all algorithms when f_n changes through AG and T. For the BCA algorithm, it is clear to see that when f_n increases, the running time increases. When $f_n = 3000$, the average gap is equal to 0.02. This means that with big-scale instances, the best-proposed algorithm provides good results. On the other hand, an average gap of 0.54 is reached for RCT when $f_n = 3000$. The maximum gap value of 0.87 is noted for ECT when $f_n = 300$.

TABLE V
RESULTS PRESENTATION FOR ALL ALGORITHMS WHEN f_n CHANGES

		f_n								
		10	30	50	100	300	500	1000	2000	3000
LFF	AG	0.31	0.73	0.69	0.58	0.51	0.38	0.31	0.36	0.31
	T	-	-	-	-	-	-	-	-	-
SCT	AG	0.09	0.06	0.06	0.10	0.08	0.07	0.23	0.02	0.02
	T	-	-	0.01	0.02	0.06	0.10	0.22	0.45	0.68
CCT	AG	0.29	0.47	0.58	0.51	0.42	0.34	0.30	0.36	0.30
	T	-	-	-	0.02	0.05	0.08	0.18	0.34	0.50
ECT	AG	0.58	0.84	0.74	0.64	0.87	0.69	0.70	0.76	0.85
	T	-	-	0.01	0.02	0.06	0.10	0.26	0.45	0.66
RCT	AG	0.00	0.29	0.36	0.36	0.19	0.17	0.05	0.14	0.54
	T	-	0.01	0.01	0.04	0.12	0.19	0.40	0.79	1.29
BCA	AG	0.00	0.04	0.04	0.10	0.04	0.05	0.03	0.01	0.02
	T	-	0.01	0.02	0.06	0.18	0.29	0.63	1.23	1.97

C. Servers number variation

Table VI depicts the results for all algorithms when s_n changes through AG and T. The AG values provided by the BCA algorithm are the lowest compared to other algorithms. For a large number of servers (i.e., $S_n = 50$, $AG = 0.02$), the BCA algorithm continues to be the best in terms of the gap. For $S_n = 5$, AG is equal to zero, which means that the BCA reaches its perfect value. In addition, we notice that for $S_n \leq 5$, the running processing time T is less than $0.01s$ for all the algorithms. This means that for a number of $S_n \leq 5$, the proposed algorithms run similarly. Beyond the value of $S_n = 5$, T increases for all the algorithms, especially for the BCA algorithm, which provides a T value of 1.64. For the algorithm LFF, the running time is always less than 0.01 s.

TABLE VI
IMPACT OF THE CHANGE OF s_n ON THE CONSIDERED ALGORITHMS

		s_n						
		3	4	5	10	20	30	50
LFF	AG	0.66	0.54	0.53	0.40	0.42	0.37	0.39
	T	-	-	-	-	-	-	-
SCT	AG	0.07	0.07	0.08	0.11	0.07	0.09	0.08
	T	-	-	-	0.18	0.23	0.27	0.63
CCT	AG	0.48	0.53	0.34	0.34	0.39	0.36	0.38
	T	-	-	-	0.11	0.17	0.22	0.53
ECT	AG	0.73	0.58	0.86	0.89	0.85	0.48	0.83
	T	-	-	-	0.20	0.24	0.27	0.61
RCT	AG	0.16	0.17	0.32	0.34	0.22	0.20	0.19
	T	0.01	0.01	0.01	0.38	0.44	0.50	1.02
BCA	AG	0.01	0.00	0.07	0.06	0.02	0.05	0.02
	T	0.01	0.01	0.01	0.55	0.68	0.77	1.64

D. Class variation

Table VII shows the impact of the change of $Class$ on the considered algorithms using AG and T. We notice a small variation in the running time T with all the algorithms. This means that the class variation does not influence T. The best values of AG are also given by the BCA algorithm. BCA algorithm has achieved the same value of AG (0.02) for classes 1, 4, and 5, and a little higher values for classes 2 and 3.

TABLE VII
RESULTS PRESENTATION FOR ALL ALGORITHMS WHEN $Class$ CHANGES

		$Class$				
		1	2	3	4	5
LFF	AG	0.37	0.43	0.44	0.50	0.51
	T	-	-	-	-	-
SCT	AG	0.08	0.13	0.11	0.05	0.06
	T	0.20	0.20	0.21	0.19	0.19
CCT	AG	0.30	0.34	0.35	0.47	0.49
	T	0.15	0.16	0.17	0.14	0.14
ECT	AG	0.84	0.85	0.86	0.59	0.60
	T	0.22	0.21	0.20	0.19	0.19
RCT	AG	0.24	0.29	0.32	0.16	0.16
	T	0.38	0.39	0.37	0.35	0.35
BCA	AG	0.02	0.06	0.05	0.02	0.02
	T	0.58	0.59	0.58	0.54	0.55

E. Algorithms comparison

Hereafter, we denote by the pair (f_n,S_n) all different values of f_n and S_n. In total, we have 30 pair values beginning with pair(10,3) and ending with pair(3000,50). Figure 6 illustrates

the comparison based on AG and pair (f_n,S_n) between the algorithms LFF and SCT. We notice that SCT outperforms LFF since the AG values are the lowest.

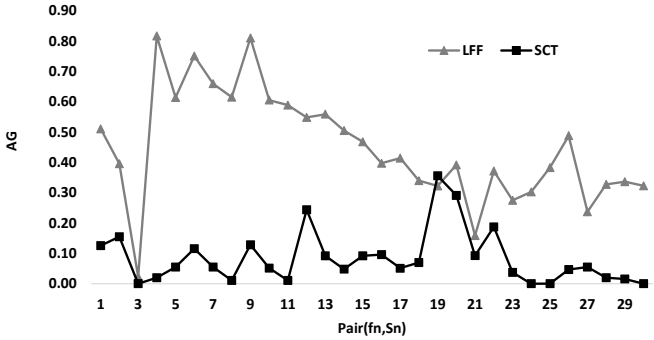

Fig. 6. Comparison based on AG and pair (f_n,S_n) between LFF and SCT algorithms

Figure 7 depicts a comparison between the algorithms CCT and ECT according to AG and pair(f_n,S_n). The variation of AG with the CCT algorithm is the lowest compared to the ECT algorithm, which is characterized by the non-stability of AG values. The curve of the ECT algorithm is above the curve of CCT. This means that CCT gives better results than ECT.

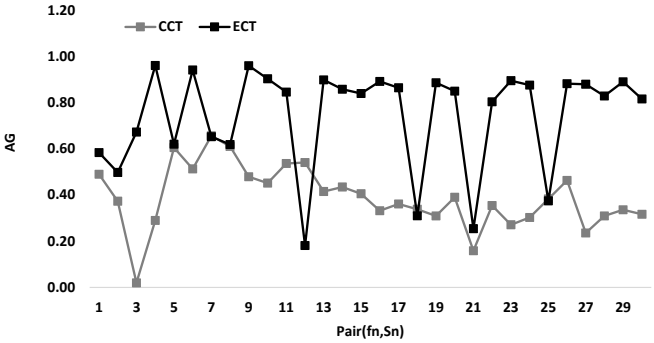

Fig. 7. Comparison between CCT and ECT algorithms based on AG and pair (f_n,S_n)

Figure 8 illustrates a comparison between RCT and BCA algorithms based on AG and pair (f_n,S_n). The BCA algorithm presents the best results compared to the RCT algorithm according to AG values.

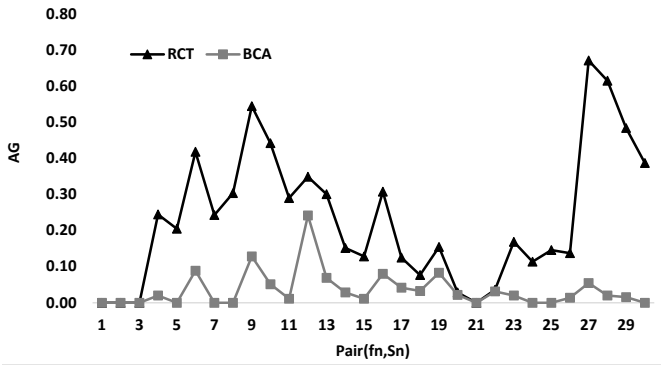

Fig. 8. Comparison between RCT and BCA algorithms according to AG and pair (f_n,S_n)

VI. Conclusion

This study introduces an advanced heuristic scheduling methodology to reduce consumer expenses while enhancing edge storage efficiency. It proposes six novel algorithms, detailing their theoretical concepts and practical applications. Extensive experiments have been conducted to evaluate the performance of these algorithms. The results indicate that these algorithms significantly contribute to lowering consumer costs. Various metrics, including gap, percentage Prg, and runtime, have been used to highlight the unique advantages and limitations of each algorithm within the edge computing context. The experiments demonstrate the superior performance of the clustering and randomization methods. The Best Clustering Algorithm (BCA) achieved an excellent efficiency rate of 92.2% and an average runtime of merely 0.04 seconds, highlighting the effectiveness of the proposed approach. Future research will further refine the proposed algorithms using innovative strategies, potentially integrating Variable Neighborhood Search (VNS) and other meta-heuristic techniques. The development of a comparative lower bound for gap evaluations is also planned. Additionally, the proposed solutions could be applied to various problems and contexts, including heterogeneous edge-cloud environments.

Data availability

The dataset used in this study is available for download from : https://www.kaggle.com/datasets/boulila/resource-management-for-cost-optimization-in-iot/.

Acknowledgements

The authors would like to thank Prince Sultan University for their support.

References

[1] M. Jemmali, M. Denden, W. Boulila, R. H. Jhaveri, G. Srivastava, and T. R. Gadekallu, "A novel model based on window-pass preferences for data-emergency-aware scheduling in computer networks," *IEEE Transactions on Industrial Informatics*, 2022.

[2] J. Li, G. Wu, T. Liao, M. Fan, X. Mao, and W. Pedrycz, "Task scheduling under a novel framework for data relay satellite network via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 5, pp. 6654–6668, 2023.

[3] Y. Song, J. Ou, P. N. Suganthan, W. Pedrycz, Q. Yang, and L. Xing, "Learning adaptive genetic algorithm for earth electromagnetic satellite scheduling," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 6, pp. 9010–9025, 2023.

[4] K. Lei, P. Guo, Y. Wang, J. Zhang, X. Meng, and L. Qian, "Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 1, pp. 1007–1018, 2024.

[5] A. Banjar, M. Jemmali, L. K. B. Melhim, W. Boulila, T. Ladhari, and A. Y. Sarhan, "Intelligent scheduling algorithms for the enhancement of drone based innovative logistic supply chain systems," *IEEE Access*, 2023.

[6] C. Zhang, F. Wu, H. Wang, H. Zhang, H. Ma, and Y. Liu, "A deep reinforcement learning model for a two-layer scheduling policy in urban public resources," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 2712–2727, 2024.

[7] G. Kumaresan, K. Devi, S. Shanthi, B. Muthusenthil, and A. Samydurai, "Hybrid fuzzy archimedes-based light gbm-xgboost model for distributed task scheduling in mobile edge computing," *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 4, p. e4733, 2023.

[8] L. Longshu and W. Qingqing, "Self-adaptive differential evolution algorithm based on opposition-based learning," *Journal of Computer Applications*, vol. 38, no. 2, p. 399, 2018.

[9] L. B. Costa, H. Yang, E. Vairavanathan, A. Barros, K. Maheshwari, G. Fedak, D. Katz, M. Wilde, M. Ripeanu, and S. Al-Kiswany, "The case for workflow-aware storage: An opportunity study," *Journal of Grid Computing*, vol. 13, no. 1, pp. 95–113, 2015.

[10] Y. Huang, J. Zhang, J. Duan, B. Xiao, F. Ye, and Y. Yang, "Resource allocation and consensus of blockchains in pervasive edge computing environments," *IEEE Transactions on Mobile Computing*, vol. 21, no. 9, pp. 3298–3311, 2021.

[11] J. Singh, P. Singh, M. Hedabou, and N. Kumar, "An efficient machine learning-based resource allocation scheme for sdn-enabled fog computing environment," *IEEE Transactions on Vehicular Technology*, 2023.

[12] F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: a survey," *The Journal of Supercomputing*, vol. 71, no. 9, pp. 3373–3418, 2015.

[13] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends," *Swarm and Evolutionary Computation*, vol. 62, p. 100841, 2021.

[14] A. Marahatta, S. Pirbhulal, F. Zhang, R. M. Parizi, K.-K. R. Choo, and Z. Liu, "Classification-based and energy-efficient dynamic task scheduling scheme for virtualized cloud data center," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1376–1390, 2019.

[15] I. M. Ali, K. M. Sallam, N. Moustafa, R. Chakraborty, M. Ryan, and K.-K. R. Choo, "An automated task scheduling model using non-dominated sorting genetic algorithm ii for fog-cloud systems," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2294–2308, 2020.

[16] R. A. Haidri, C. P. Katti, and P. C. Saxena, "Capacity based deadline aware dynamic load balancing (cpdalb) model in cloud computing environment," *International Journal of Computers and Applications*, vol. 43, no. 10, pp. 987–1001, 2021.

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

[18] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in iaas cloud computing environment," *PloS one*, vol. 12, no. 5, p. e0176321, 2017.

[19] D. Luo, J. Liu, and Z. Xin, "Heuristic scheduling algorithm for hybrid storage data in the cloud computing environment," *International Journal of Internet Protocol Technology*, vol. 13, no. 3, pp. 131–136, 2020.

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

[21] M. Jemmali, A. K. Bashir, W. Boulila, L. K. B. Melhim, R. H. Jhaveri, and J. Ahmad, "An efficient optimization of battery-drone-based transportation systems for monitoring solar power plant," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–9, 2022.

[22] D. Mohapatra and B. Subudhi, "Development of a cost-effective iot-based weather monitoring system," *IEEE Consumer Electronics Magazine*, vol. 11, no. 5, pp. 81–86, 2022.

[23] X. Yu, M. Youill, M. Woicik, A. Ghanem, M. Serafini, A. Aboulnaga, and M. Stonebraker, "Pushdowndb: Accelerating a dbms using s3 computation," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1802–1805.

[24] P. Bryk, M. Malawski, G. Juve, and E. Deelman, "Storage-aware algorithms for scheduling of workflow ensembles in clouds," *Journal of Grid Computing*, vol. 14, no. 2, pp. 359–378, 2016.

[25] 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, 2015.

[26] E. N. Al-Khanak, S. P. Lee, S. U. R. Khan, N. Behboodian, O. I. Khalaf, A. Verbraeck, and H. van Lint, "A heuristics-based cost model for scientific workflow scheduling in cloud," *CMC Comput. Mater. Contin*, vol. 67, pp. 3265–3282, 2021.

[27] S. Meng, W. Huang, X. Yin, M. R. Khosravi, Q. Li, S. Wan, and L. Qi, "Security-aware dynamic scheduling for real-time optimization in cloud-based industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4219–4228, 2020.

[28] Y. Chen, L. Wang, X. Chen, R. Ranjan, A. Y. Zomaya, Y. Zhou, and S. Hu, "Stochastic workload scheduling for uncoordinated datacenter clouds with multiple qos constraints," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1284–1295, 2016.

[29] C. González García, E. R. Núñez Valdéz, V. García Díaz, B. C. Pelayo García-Bustelo, J. M. Cueva Lovelle *et al.*, "A review of artificial intelligence in the internet of things," *International Journal Of Interactive Multimedia And Artificial Intelligence, 5*, 2019.

[30] A. Aral, I. Brandic, R. B. Uriarte, R. De Nicola, and V. Scoca, "Addressing application latency requirements through edge scheduling," *Journal of Grid Computing*, vol. 17, pp. 677–698, 2019.

[31] S. Chen, Q. Li, M. Zhou, and A. Abusorrah, "Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm," *Sensors*, vol. 21, no. 3, p. 779, 2021.

[32] R. Luo, H. Jin, Q. He, S. Wu, and X. Xia, "Cost-effective edge server network design in mobile edge computing environment," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 4, pp. 839–850, 2022.

[33] H. Mohammadi Rouzbahani, H. Karimipour, and L. Lei, "Optimizing resource swap functionality in ioe-based grids using approximate reasoning reward-based adjustable deep double q-learning," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 3, pp. 522–532, 2023.

[34] X. Zhou, Y. Zeng, Z. Wu, and J. Liu, "Distributed optimization based on graph filter for ensuring invariant simplification of high-volume point cloud," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 3, pp. 608–621, 2023.

[35] H. R. Chi, M. de Fátima Domingues, H. Zhu, C. Li, K. Kojima, and A. Radwan, "Healthcare 5.0: In the perspective of consumer internet-of-things-based fog/cloud computing," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 745–755, 2023.

[36] R. Zheng, K. Huang, H. Shen, and L. Ma, "Continuous volumetric convolution network with self-learning kernels for point clouds," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 2, pp. 148–155, 2022.

[37] C. Chen, H. Lu, H. Hong, H. Wang, and S. Wan, "Deep self-supervised graph attention convolution autoencoder for networks clustering," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 974–983, 2023.

[38] S. Basu, D. Bera, and S. Karmakar, "Detection and intelligent control of cloud data location using hyperledger framework," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 1, pp. 76–86, 2022.

[39] H. Yong, J. Lee, and J.-S. Kim, "Design and implementation of virtual stream management for nand flash-based storage," *IEEE Transactions on Consumer Electronics*, vol. 67, no. 2, pp. 149–157, 2021.

[40] A. Ghosh, D. Chakraborty, and A. Law, "Artificial intelligence in internet of things," *CAAI Transactions on Intelligence Technology*, vol. 3, no. 4, pp. 208–218, 2018.

[41] S. Mahapatra, R. R. Dash, and S. K. Pradhan, "Heuristics techniques for scheduling problems with reducing waiting time variance," *Heuristics and Hyper-Heuristics-Principles and Applications*, pp. 43–64, 2017.

[42] W. Boulila, "Resource Management for Cost Optimization in IoT," https://www.kaggle.com/datasets/boulila/resource-management-for-cost-optimization-in-iot/, 2024, [Online; accessed 23-April-2024].