Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.040919

ARTICLE

Intelligent Solution System for Cloud Security Based on Equity Distribution: Model and Algorithms

Sarah Mustafa Eljack[1,*], Mahdi Jemmali[2,3,4], Mohsen Denden[6,7], Mutasim Al Sadig[1], Abdullah M. Algashami[1] and Sadok Turki[5]

[1]Department of Computer Science and Information, College of Science at Zulfi, Majmaah University, Al-Majmaah, 11952, Saudi Arabia

[2]Department of Computer Science, College of Computing and Informatics, University of Sharjah, Sharjah, United Arab Emirates

[3]Higher Institute of Computer Science and Mathematics, University of Monastir, Monastir, 5000, Tunisia

[4]Mars Laboratory, University of Sousse, Sousse, Tunisia

[5]Department of Logistic and Maintenance, UFR MIM at Metz, University of Lorraine, Metz, France

[6]Department of Computer and Information Technologies, College of Telecommunication, and Information Riyadh CTI, Technical and Vocational Training Corporation TVTC, Riyadh, 12464, Saudi Arabia

[7]Department of Computer Science, Higher Institute of Applied Sciences of Sousse, Sousse University, Sousse, 4000, Tunisia

*Corresponding Author: Sarah Mustafa Eljack. Email: s.alshiekh@mu.edu.sa

Received: 04 April 2023 Accepted: 03 November 2023 Published: 30 January 2024

ABSTRACT

In the cloud environment, ensuring a high level of data security is in high demand. Data planning storage optimization is part of the whole security process in the cloud environment. It enables data security by avoiding the risk of data loss and data overlapping. The development of data flow scheduling approaches in the cloud environment taking security parameters into account is insufficient. In our work, we propose a data scheduling model for the cloud environment. The model is made up of three parts that together help dispatch user data flow to the appropriate cloud VMs. The first component is the Collector Agent which must periodically collect information on the state of the network links. The second one is the monitoring agent which must then analyze, classify, and make a decision on the state of the link and finally transmit this information to the scheduler. The third one is the scheduler who must consider previous information to transfer user data, including fair distribution and reliable paths. It should be noted that each part of the proposed model requires the development of its algorithms. In this article, we are interested in the development of data transfer algorithms, including fairness distribution with the consideration of a stable link state. These algorithms are based on the grouping of transmitted files and the iterative method. The proposed algorithms show the performances to obtain an approximate solution to the studied problem which is an NP-hard (Non-Polynomial solution) problem. The experimental results show that the best algorithm is the half-grouped minimum excluding (HME), with a percentage of 91.3%, an average deviation of 0.042, and an execution time of 0.001 s.

KEYWORDS

Cyber-security; cloud computing; cloud security; algorithms; heuristics

1 Introduction

With the improvement of computer and communication technology and the increasing need for human quality of life, intelligent devices are growing in popularity. Internet applications are growing in diversity and complexity due to the development of artificial intelligence algorithms and communication technologies. Traditional cloud computing, which is used to support general computing systems, can hardly satisfy the needs of IoT (Internet of Things) and mobile services due to location unawareness, bandwidth shortage, operation cost imposition, lack of real-time services, and lack of data privacy guarantee. These limitations of cloud computing pave the way for the advent of edge computing. This technology is believed to cope with the demands of the ever-growing IoT and mobile devices. The basic idea of edge computing is to employ a hierarchy of edge servers with increasing computation capabilities to handle mobile and heterogeneous computation tasks offloaded by the low-end (IoT) and mobile devices, namely edge devices. Edge computing has the potential to provide location-aware, bandwidth-sufficient, real-time, privacy-savvy, and low-cost services to support emerging innovative city applications. Such advantages over cloud computing have made edge computing rapidly grow in recent years. Cloud computing consists of many data centers that house many physical machines (hosts). Each host runs multiple virtual machines (VMs) that perform user tasks with different quality of service (QoS). Users can access cloud resources using cloud service providers on a pay-as-you-go basis.

The IoT environment associated with the cloud computing paradigm makes efficient use of already available physical resources thanks to virtualization technology. Thus, multiple healthcare service users (HCS) can store and access various healthcare resources using a single physical infrastructure that includes hardware and software. One of the most critical problems in healthcare services is the task scheduling problem. This problem causes a delay in receiving medical requests in the healthcare service by users in cloud computing environments.

In this work, a new model was developed to store user data with fair distribution in cloud virtual machines. The used method can reinforce the security of the stored data. Two types of information are distinguished in this model. The first type is the data flow generated by users. It is random data because time and size are unknown. The second one is the control information or mapping information gathered by the collector agent and analyzed by the monitor agent according to the security levels. The scheduler receives the random user data which represents the main input for our algorithms and the regular information from the monitor agent which represents the second entree parameters for six algorithms. In this paper, the two presented agents are proposed in the model description and explained as part of our global model. The interaction of these agents with the scheduler will be treated in future work. This paper focused on the development of algorithms for equity distribution. For each data flow, developed algorithms should indicate the appropriate virtual machine and ensure a fair distribution of all incoming data. The task scheduling algorithms in the literature are used to reach an objective like minimizing the Makespan or latency or other well-known objectives. In this paper, a new objective is proposed. In addition, novel algorithms based on the grouping method are developed and assessed to show their performance. Consequently, the proposed algorithms can be reformulated and applied to solve traditional scheduling problems like parallel machines flow shops, or other hard problems. The main contributions of this paper are:

- Developing a new model for efficient file storage.
- Develop algorithms for equity distribution related to the virtual machines in the cloud environment.
- Minimize the risk of losing data by ensuring an equity distribution for the virtual machines.

- Compare the efficiency of the proposed algorithms and their complexity.

This paper is structured as follows. Section 2, is reserved for the related works. Section 3 presents the study background. In Section 4, the proposed algorithms are detailed and explained. The experimental results are discussed in Section 5. Finally, the conclusion is given in Section 6.

2 Related Works

Some classical scheduling techniques, such as first-come-first-served (FCFS), round robin (RR), and shortest job first (SJF), can provide scheduling solutions. Still, the scheduling problem is NP-hard, which makes cloud computing difficult. It fails to meet the needs of programming [1]. Since traditional scheduling algorithms cannot solve NP-hard optimization problems, modern optimization algorithms, also called meta-heuristic algorithms, are used nowadays instead. These algorithms can generate optimal or near-optimal solutions in a reasonable time compared to traditional planning algorithms. Several metaheuristic algorithms have been used to deal with task scheduling in cloud computing environments. For example, a new variant of conventional particle swarm optimization (PSO), namely his PSO (RTPSO) based on ranging and tuning functions, was proposed in [2] to achieve better task planning. In RTPSO, the inertia weight factors are improved to generate small and large values for better local search and global searches. RTPSO was merged into the bat algorithm for further improvement. This variant he named RTPSO-B.

In [3], the authors developed a task scheduling algorithm for bi-objective workflow scheduling in cloud computing based on a hybrid of the gravity search algorithm (GSA) and the heterogeneous earliest finish time (HEFT). This algorithm he called HGSA. This algorithm is developed to reduce manufacturing margins and computational costs. However, GSA sometimes does not work accurately for more complex tasks. The bat algorithm (BA) is applied to address the task scheduling problem in cloud computing with objective features to reduce the overall cost of the workflow [4,5]. On the other hand, BA underperformed in higher dimensions. Several papers treated load balancing in different domains. In finance and budgeting [6–9], storage systems [10], smart parking [11], the network [12], and parallel machines [13]. The authors in [14] proposed two variants of PSO. The first, called LJFP-PSO, is based on initializing the population using a heuristic algorithm known as the longest job to fastest processor (LJFP). On the other hand, the second variant, MCT-PSO, uses the MCT (minimum completion time) algorithm to initialize the population and improve the manufacturing margin, total execution time, and non-uniformity when dealing with task scheduling problems in the cloud.

This answer intended to limit the general execution value of jobs, at the same time as maintaining the whole of entirety time inside the deadline [15]. According to the findings of a simulation, the GSO primarily based mission scheduling (GSOTS) set of rules has higher consequences than the shortest mission first (STF), the most essential mission first (LTF), and the (PSO) algorithms in phrases of reducing the whole of entirety time and the value of executing tasks. There are numerous different metaheuristics-primarily based mission scheduling algorithms inside the cloud computing environment, which include the cuckoo seek a set of rules (CSA) [16], electromagnetism optimization (EMO) set of rules [17], sea lion optimization (SLO) set of rules [18], adaptive symbiotic organisms seek (ASOS) [19], hybrid whale optimization set of rules (HWOA) [20], synthetic vegetation optimization set of rules (AFOA) [21], changed particle swarm optimization (MPSO) [22], and differential evolution (DE) [23–27]. In the same context, other research works are developed [28–30].

The algorithms developed in this paper can be extended to be used for the subject treated in [31–35]. The techniques for machine learning and deep learning can be utilized to develop new

algorithms for the studied problem [36]. The Prevention Mechanism in Industry 4.0, the vehicular fog computing algorithms, and the Scheme to resist denial of service (DoS) can also be adapted to the novel constraint of the proposed problem [37–40].

According to literature research. Most of the heuristics and approaches developed to schedule data flows focus on minimizing processing time and optimizing resources. There is not much research that integrates the scheduling problem and the data security problem. By developing our approach, we believe we can integrate certain security parameters into the scheduling algorithms. We believe that the combination of the two techniques saves data processing time. In this article, we succeeded in developing the best scheduling algorithms by considering the security parameters (Collector and Monitor Agents) as constants. Our work continues to develop selection algorithms (trusted paths) and integrate them with those recently developed.

In general, Cloud environments can have many internal and external vulnerabilities such as Imminent risk linked to a bad configuration, possibly bad partnership/deployment strategy, and unauthorized access to resources, which increases the attack surface. To keep this environment accessible and secure, access controls, multi-factor authentication, data protection, encryption, configuration management, etc., are essential. Our goal is to place cloud data streams in the right places while minimizing data security risks.

3 Study Background

Nowadays, the cloud environment has become the primary environment to run applications that require large capacities. Several areas use the cloud because of its scalability, fast services, and "we pay for what we consume". Data flow planning in the cloud aims to minimize the overall execution time. It consists of building a map of all the necessary components to achieve tasks from source to destination. This process is called task mapping. The cloud environment faces many challenges, such as cost and power consumption reduction of various services. The optimization of the cost is studied in [41–44]. Recently, many international companies have ranked security [45–47] as the most difficult parameter to achieve in the cloud and the first challenge for cloud developers and users. In the literature, several heuristics and meta-heuristics have been developed to optimize cost, processing time, and distribute workflows and tasks [48–50]. Heuristics and algorithms for planning workflow in a cloud environment, taking into account security parameters, are not extensively developed. The security process includes an additional process time. The objective of our research is to develop new heuristics to reduce unused or mismanaged network resources. The developed model consists of periodically collecting information on the state of links, and intermediate nodes in the network, using a collector agent. This information is analyzed and subsequently classified using a monitor agent according to the level of confidence. The novel model is based essentially on three agents, as illustrated in Fig. 1.

"**Collector Agent**" has the role of the collector to gather all possible information on the global state of the network. The collector agent scans all available resources and collects information about the state of the cloud virtual machine and the network link. This component translates this information to the monitor which processes it, analyzes it and decides on the security level of each link, and transmits it to the scheduler.

The information generated is an additional input for the scheduler which must take it into account in each calculation. Several other works can be considered [51–53].

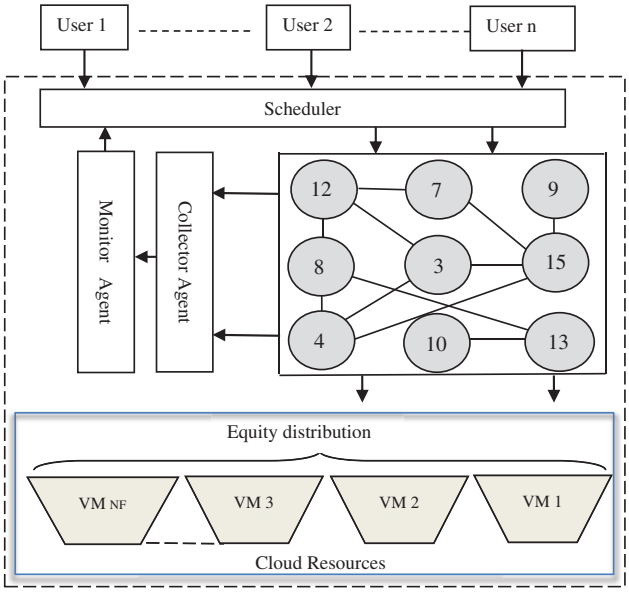

Figure 1: Model for the cloud computing environment

Information related to the security level will be treated specially. The second agent is the "Monitoring agent" which must analyze received information, estimate the risk levels on each section of the network, and decide whether the final state of the network is a downlink, low-risk or fair-link, or high-risk link (See Fig. 2). The Scheduler component run the best-proposed algorithm to make a final decision about workflow dispatching. Users generate an arbitrary size of data flow. Finally, the "Cloud resources" component represents the set of virtual machines (storage servers or Hosts, etc.).

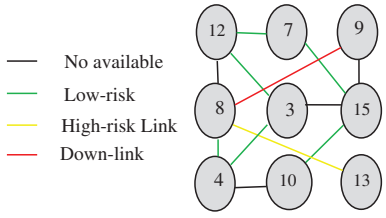

Figure 2: Monitoring agent decomposition

The scheduler should define the destination of each file ensuring the load balancing of the final state of the virtual machines. The network path state is described in Fig. 2. In this work, we focus on the role of the scheduler component that calls the best-proposed algorithm to solve the problem of dispatching files that must be executed by the virtual machines. The two presented agents are proposed in the model description and explained as part of our global model. Algorithms for the two agents will be treated in future work. In this paper, the objective is to propose several algorithms to solve the scheduling problem of transmission of files to different virtual machines ensuring equity distribution. To do that, the gap of the used spaces between all virtual machines must be searched. This gap is

denoted by GpV and given in Eq. (1):

$$GpV = \sum_{i=1}^{Nv} (Tv_i - Tv_{min}) \tag{1}$$

The variable descriptions are presented in Table 1. The objective is the minimize GpV. This problem is already proved as an NP-hard problem in [10].

Table 1: Description of all used notations

Notation	Description
F	Set of given files
Nf	Number of files
Nv	Number of virtual machines
Vm	Set of virtual machines
j	Index of each file
i	Index of a virtual machine
V_i	Virtual machine number i
f_j	File number j
sf_j	Size of the file f_j
tf_j	Used space when the file f_j is transmitted
Tv_i	Total space in the virtual machine V_i
Tv_{min}	Minimum used space $\forall V_i, i = \{1, \ldots, Nv\}$

Example 1 may clarify the objective proposed in this paper.

Example 1

Assume that eight files must be transmitted to three virtual machines. In this case, $Nf = 8$ and $Nv = 3$. The size files sf_j for the file $f_j \forall j = \{1, \ldots, Nf\}$ are presented in Table 2.

Table 2: Size of the eight files

j	1	2	3	4	5	6	7	8
sf_j	16	7	9	11	8	21	6	12

This example used two algorithms to explain the proposed problem. The first algorithm is the shortest size first (SSF). This algorithm is based on the sorting of all files according to the increasing order of their size. After that, the scheduling of the files will be done one by one on the virtual machine that has the minimum Tv_i. The second algorithm is the longest-size file algorithm (LSF). Firstly, the files are sorted according to the decreasing order of their sizes. Next, the first file will be assigned to the first virtual with the minimum Tv_i value and so on, until the scheduling of all files. For SSF, files {1, 3, 7} are transmitted to the first virtual machine, files {2, 4, 6} are transmitted to the second virtual machine, and files {5, 8} are transmitted to the second virtual machine. Consequently, the load Tv_1 in V_1, Tv_2 in V_2, and Tv_3 in V_3 are 31, 39, and 20, respectively. Thus, $Tv_{min} = 20$. So, the GpV value is $\sum_{i=1}^{3}(Tv_i - Tv_{min}) = (31 - 20) + (39 - 20) + (20 - 20) = 30$. In this case, the gap between

the used spaces between all virtual machines is 30. Now, it is crucial to find another schedule that gives a better solution than the latter result. This means that find a schedule with a minimum value of GpV. In this context, the algorithm LSF is applied. This algorithm gives the schedule as follows: files {5, 6} are transmitted to the first virtual machine, files {1, 3, 7} are transmitted to the second virtual machine, and files {2, 4, 8} are transmitted to the second virtual machine. Consequently, the load Tv_1 in V_1, Tv_2 in V_2, and Tv_3 in V_3 are 29, 31, and 30, respectively. Thus, $Tv_{min} = 29$. So, the GpV value is $\sum_{i=1}^{3}(Tv_i - Tv_{min}) = (29 - 29) + (31 - 29) + (30 - 29) = 3$. In this case, the gap of the used spaces between all virtual machines is 3. This schedule is better than the first one. A gain of 27 units is reached after applying the second algorithm.

To ensure a fair distribution, the algorithms must always calculate the difference in capacity between the virtual machines also called the gap (the gap is the unused space in each virtual machine). The main role of algorithms is to minimize the gap (capacity) between different virtual machines. The scheduler must then transfer a flow of capital data to the appropriate virtual machine (the one which must minimize the difference between the different virtual machines). In this way, data stability is more secure and several tasks that may occur such as data migration are avoided. Data stability means even less risk. On the other hand, if we consider that the states of the links are stable, this means that there is no other constraint to be taken into account by the scheduler, it must ensure a fair distribution in the first place. If the link states are not stable, other factors must be considered in the calculation of the scheduler since the virtual machines will not be "judged" according to their level of use but also according to the level of risk. This is what we are currently developing.

The main idea is to forward the data stream to the appropriate cloud virtual machines. In the developed model, two new components are introduced which are the collector agent and the monitor agent. The collector agent collects information about the state of the network (nodes and links), mainly security information (denial of service, downlinks, virtual machines at risk, etc.). This information will be transmitted to the surveillance agent who traits this information, decides the risk level degree, and transmits it to the scheduler. The scheduler must consider the decision of the monitor agent as input and re-estimate its new decision according to the new considerations.

4 Proposed Algorithms

In this section, several algorithms to solve the studied problem are proposed. These algorithms are based on the grouping method. This method consists of dividing the set of files into two groups. After that, several manners and variants are applied to choose how to schedule the files on the virtual machines between the first group and the second one. Essentially, seven-based algorithms are proposed. The proposed algorithms are executed by applying four steps. Fig. 3 shows the steps of the objective reached by algorithms. The first step is the "Objective" which is the load-balancing schedule. The second step is the mathematical formulation to reach the load balancing. The third step is minimizing the load gap by reaching the solution. The fourth step is the analysis of the performance of each solution obtained by the proposed algorithms.

4.1 Longest Size File Algorithm (LSF)

Firstly, the files are sorted according to the decreasing order of their sizes. Next, the first file will be assigned to the first virtual machine with minimum Tv_i value, and so on, until the scheduling of all files. The complexity of the algorithm is $O(nlogn)$.

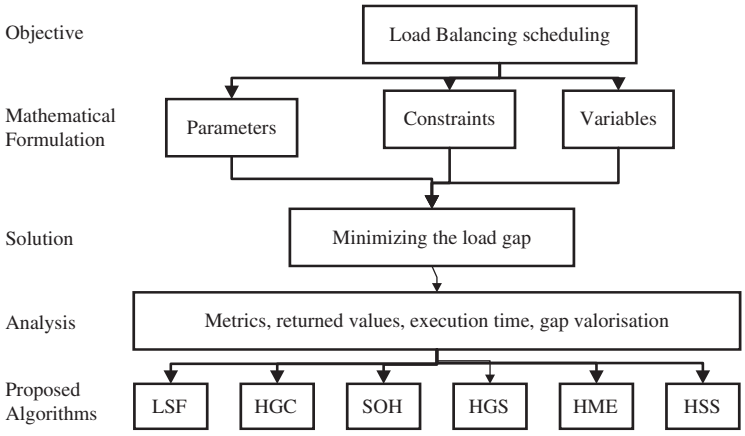

Figure 3: Steps of the objective reached by algorithms

4.2 Half-Grouped Classification Algorithm (HGC)

This algorithm is based on dividing the set of files into two groups. Initially, the two groups are empty. The number of files in the first group, G_1 is $n_1 = \left\lfloor \frac{Nf}{2} \right\rfloor$. While the number of files in the second group, G_2 is $n_2 = Nf - n_1$. In fact, $G_1 = \{f_1, \ldots, f_{n_1}\}$ et $G_2 = \{f_{n_1+1}, \ldots, f_{Nf}\}$ where $F = \{f_1, \ldots, f_{Nf}\}$. The proposed algorithm uses three variants. For each variant, a solution is calculated, and the best solution is picked. In the first variant, there any sort of changes to the set F is made. After creating the two groups, G_1 and G_2, two manners are applied to scheduling files. The first is to schedule all files in G_1. After that the files in G_2 are scheduled. The second manner is to schedule all files in G_2. After that, all files in G_1 are scheduled. The solution is calculated for this first variant. In the second variant, the files in F are sorted according to the decreasing order of their sizes. After creating the two groups, G_1 and G_2, two manners are applied to scheduling files. The first is to schedule all files in G_1. After that, all files in G_2 are scheduled. The second manner is to schedule all files in G_2. After that, all files in G_1 are scheduled. The solution is calculated for this first variant. In the third variant, the files in F are scheduled according to the increasing order of their sizes. After creating the two groups, G_1 and G_2, two manners are applied to scheduling files. The first is to schedule all files in G_1. After that, all files in G_2 are scheduled. The second manner is to schedule all files in G_2. After that, all files in G_1 are scheduled. The solution is calculated for this first variant. Denotes by $DCs\,(A)$, the procedure that accepts a set of elements as input and sorts these elements according to the decreasing order of their values. While $ICs(A)$ is the procedure that accepts a set of elements as input and sorts them according to the decreasing order of their values. Procedure $Sh(A)$ schedules the elements on the virtual machines one by one. The virtual machine is selected to schedule an element characterized by the minimum value of Tv_i. The procedure of each variant denoted by $HGCP()$ is detailed in Algorithm 1. This procedure returns the solutions GpV_k^1 and GpV_k^2 with $k = \{1, 2, 3\}$. The execution details of HGC are given in Algorithm 2. The complexity of the algorithm is $O(nlogn)$.

Algorithm 1: Half-Grouped Classification Procedure Algorithm (*HGCP*)

***Step*0**	Construct G_1 and G_2
***Step*1**	Call $Sh(G_1)$
***Step*2**	Call $Sh(G_2)$
***Step*3**	Calculate GpV_k^1
***Step*4**	Call $Sh(G_2)$
***Step*5**	Call $Sh(G_1)$
***Step*6**	Calculate GpV_k^2

Algorithm 2: Half-Grouped Classification Algorithm (*HGC*)

***Step*0**	**for** (three variants) **do**
***Step*1**	Call $HGCP(F)$
***Step*2**	**end for**
***Step*3**	Calculate $GpV = \min_{1 \le k \le 3}(GpV_k^1, GpV_k^2)$

4.3 Mini-Load Half-Grouped Algorithm (MLH)

This algorithm is based on the grouping method. The two groups, G_1 and G_2 are constructed by the same method detailed in the above algorithm (Step 1 in Algorithm 3). The proposed algorithm uses three variants. For each variant, a solution is calculated, and the best solution is picked. In the first variant, there is any sort of changes to the set F. After creating the two groups, G_1 and G_2, the scheduling of files will be based on the minimum load of the two groups. Let "a load of a group" is the sum of all file sizes in the group. So, initially, after constructing the two groups, the load of G_1 denotes by L_{G1} is $\sum_{j=1}^{n_1} sf_j$, and the load of G_2 denotes by L_{G2} is $\sum_{j=n_1+1}^{Nf} sf_j$. If $L_{G1} \geq L_{G2}$ the first file in G_1 is selected and scheduled (Step 4 in Algorithm 3). Otherwise, the first file in G_2 is selected and scheduled (Step 7 in Algorithm 3) and soon on until all files are scheduled. The solution in this manner is calculated and denoted by GpV_1 (Step 11 in Algorithm 3). In the second variant, the files are sorted into F according to the decreasing order of their sizes. The two groups are created and apply the minimum load to choose between groups. The solution in this manner is calculated and denoted by GpV_2 (Step 13 in Algorithm 3). In the second variant, the files are sorted into F according to the increasing order of their sizes. Two groups are created and apply the minimum load to choose between groups. The solution in this manner is calculated and denoted by GpV_3 (Step 15 in Algorithm 3). Denotes by $ShF(G)$ the procedure that schedules the first file in the set G. The execution details of *MLH* are given in Algorithm 3. The complexity of the algorithm is $O(nlogn)$.

Algorithm 3: Mini-Load Half-Grouped Algorithm (*MLH*)

***Step*0**	**for** (three variants) **do**
***Step*1**	Construct G_1 and G_2
***Step*2**	**for** ($j = 1$ to Nf) **do**
***Step*3**	Calculate L_{G1} and L_{G2}
***Step*4**	**if** ($L_{G1} \geq L_{G2}$) **then**
***Step*5**	Call $ShF(G_1)$
***Step*6**	**else**
***Step*7**	Call $ShF(G_2)$

(Continued)

Algorithm 3 (continued)

***Step*8** **end if**
***Step*9** **end for**
***Step*10** **if** (*variant*1) **then**
***Step*11** Calculate GpV_1
***Step*12** **else if** (*variant*2) **then**
***Step*13** Calculate GpV_2
***Step*14** **else if** (*variant*3) **then**
***Step*15** Calculate GpV_3
***Step*16** **end if**
***Step*17** **end for**
***Step*18** Calculate $GpV = \min_{1 \le k \le 3} GpV_k$

4.4 Excluding the Nv-Files Mini-Load Half-Grouped Algorithm (ENM)

This algorithm is based on the grouping method. Firstly, the Nv big files are excluded. These Nv files denoted by LNV are scheduled on the virtual machines. Each file will be assigned to an available virtual machine. After that, for the remaining $Nf - Nv$ files denoted by RFV, the MLH algorithm is adopted to schedule these remaining files to the virtual machines taking into consideration the Nv files already scheduled. The complexity of the algorithm is $O(nlogn)$.

4.5 One-by-One Half-Grouped Algorithm (OHG)

This algorithm is based on dividing the set of files into two groups following the same way described in HGC. Three variants are used. For each variant, a solution is calculated, and the best solution is picked. In the first variant, there is no sort of changes to the set F. After creating G_1 and G_2, two manners are applied to scheduling files. The first is to schedule the first file in G_1 after that, the first file in G_2, until the scheduling of all files and the solution GpV_1 is calculated. In a second manner, all files in G_2 are scheduled. After that, all files in G_1 are scheduled. The solution GpV_2 is calculated. The Minimum between GpV_1 and GpV_2 constitutes the solution of the first variant. In the second variant, the files in F are sorted according to the decreasing order of their sizes. After creating G_1 and G_2, the two previous manners are applied to scheduling files. The solution is calculated for this second variant. In the third variant, the files in F are sorted according to the increasing order of their sizes. After creating G_1 and G_2, the two previous manners are applied to scheduling files. The solution is calculated for this third variant. The best of these three solutions is picked. The OHG instructions are given in Algorithm 5. The complexity of the algorithm is $O(nlogn)$.

Algorithm 4: One-by-One Half-Grouped Algorithm (OHG)

***Step*0** **for** (three variants) **do**
***Step*1** Construct G_1 and G_2
***Step*2** **for** ($j = 1$ to Nf) **do**
***Step*3** Call $ShF(G_1)$
***Step*4** Call $ShF(G_2)$
***Step*5** **end for**
***Step*6** Calculate GpV_k^1
***Step*7** Call $Sh(G_2)$

(Continued)

Algorithm 4 (continued)

***Step*8**	Call $Sh(G_1)$
***Step*9**	Calculate GpV_k^2
***Step*10**	**end for**
***Step*11**	Calculate $GpV = \min_{1 \le k \le 3}(GpV_k^1, GpV_k^2)$

4.6 Swap Two-Files Half-Grouped Algorithm (STH)

This algorithm is based on dividing the set of files into two groups. Three variants are used. For each variant, a solution is calculated, and the best solution is picked. In the first variant, there is no sort of changes to the set F. The two groups are created in the same way described in HGC. After that, the first file in G_1 denotes by $F1$ is swapped with the first file in G_2 denotes by $F2$. The swap is to move $F1$ to the first position in G_2 and to move $F2$ to the last position in G_1. The two manners described in HGC are applied to calculate the first solution. In the second variant, the files in F are sorted according to the decreasing order of their sizes. After creating the groups, G_1 and G_2, the first file in G_1 is swapped with the first file in G_2. Next, the two manners are applied, and the second solution is calculated. In the third variant, the files in F are sorted according to the increasing order of their sizes. The two groups are created in the same way described in HGC. After that, the first file in G_1 is swapped with the first file in G_2. The two manners described in HGC are applied to calculate the third solution. The best of these three solutions is picked. The complexity of the algorithm is $O(nlogn)$.

4.7 Swap One-in-Tenth-Files Half-Grouped Algorithm (SOH)

This algorithm is based on dividing the set of files into two groups. Three variants are used. For each variant, a solution is calculated, and the best solution is picked. In the first variant, there is no sort of changes to the set F. The two groups are created in the same way described in HGC. Let $St = \frac{Nf}{10}$. After that, the St first files in G_1 is swapped with the St first files in G_2. The swap is to move the St first files in G_1 to the front in G_2 and to move the St first files in G_2 to the last positions in G_1. The two manners described in HGC are applied to calculate the first solution. In the second variant, the files in F are sorted according to the decreasing order of their sizes. After creating the groups, G_1 and G_2, the St first files in G_1 is swapped with the St first files in G_2. Next, the two manners are applied, and the second solution is calculated. In the third variant, the files in F are sorted according to the increasing order of their sizes. The two groups are created in the same way described in HGC. After that, the St first files in G_1 is swapped with the St first files in G_2. the two manners described in HGC are applied to calculate the third solution. The best of these three solutions is picked. The procedure $SWPT(G_1, G_2)$ swaps the St files as described above. The complexity of the algorithm is $O(nlogn)$. In the experimental results, let HGS be the algorithm that returns the minimum value after the execution of HGC and SOH. In addition, let HME be the algorithm, which returns the minimum value after the execution of HGC, MLH, ENM, OHG, and STH. Finally, let HSS be the algorithm, which returns the minimum value after the execution of HGC, STH, and SOH.

5 Experimental Results

The discussion of experimental results is presented in this section. The proposed algorithms are coded in C++ and compared between them to show the best algorithm among all. The computer used that executes all programs is an i5 processor and eight memories. The operating system installed

in this later computer is Windows 10. the proposed algorithms are tested through four classes of instances. These classes are based on the normal distribution $N[x, y]$ and the uniform distribution $U[x, y]$ [54]. The different values of the file-size sf_j are given as follows: $C1$: $U[15, 150]$, $C2$: $U[90, 350]$, $C3$: $N[250, 30]$, and $C4$: $N[350, 90]$. For each pair of Nf and Nv and each class, ten instances are generated. The values of the Nf and Nv are detailed in Table 3. In total, there 1240 instances.

Table 3: Size files and virtual machines values

Nf	Nv
10, 25, 40	3, 4, 5
50, 150, 250, 350	3, 4, 5, 10
400, 500, 600	5, 10

The metrics used in [10] to measure the performance of the developed algorithms are:

- $\overbrace{F}$ is the minimum value of GpV for all algorithms.
- F is the GpV value returned by the presented algorithm.
- Pc is the percentage of instances when $\overbrace{F} = F$.
- $Ga = \dfrac{F - \overbrace{F}}{F}$ is the gap between the presented algorithm and the best-obtained value. If $F = 0$, $Ga = 0$.
- AP is the average of Ga for a group of instances.
- $Time$ is the average execution time. The symbol "+" is marked if the execution time is less than 0.001 s. The time is given in seconds.

Table 4 shows the overall results measuring the percentage, the average gap, and the time. In this latter table, the best algorithm is HME, with a percentage of 91.3%, an average gap of 0.042, and a running time of 0.001 s. The second-best algorithm is HSS, with a percentage of 81%, an average gap of 0.043, and a running time of 0.001 s. The algorithm that gives the minimum percentage of 33.4% is SOH. Table 4 shows that for all algorithms the average gap is less or equal to 0.001 s. The execution time is very close for all algorithms.

Table 4: Overall results measuring the percentage, the average gap, and the time

	LSF	HGC	SOH	HGS	HME	HSS
Pc %	56.2%	58.1%	33.4%	70.6%	91.3%	81.0%
Ga	0.174	0.159	0.376	0.082	0.042	0.043
$Time$ in seconds	+	+	+	0.001	0.001	0.001

The load balancing applied on the cloud environment solving the studied problem is not integrally used in the literature. However, the load of files through several storage supports is treated in [10]. The best algorithm developed in this latter work is $SIDA^r$. After coding this algorithm and executing it over the instances used in this paper, a comparison with the best-proposed algorithm HME results in Table 5. This latter table shows that HME gives better results than $SIDA^r$ in 23.5% of cases with 292 instances. However, $SIDA^r$ gives better results than HME in 29.3% of cases with 363 instances.

Finally, there are 585 instances when HME and $SIDA^r$ obtained the same results. To summarize, the best algorithm in [10] does not dominate the best-proposed algorithms. Consequently, a new algorithm can be developed based on the combination of HME and $SIDA^r$.

Table 5: Comparison of the best-proposed algorithm with the best from the literature

Condition	Number of instances	Percentage
$HME < SIDA^r$	292	23.5%
$HME > SIDA^r$	363	29.3%
$HME = SIDA^r$	585	47.2%

Table 6 shows the average gap values when the number of files changes for all algorithms. There are only four cases when the average gap is less than 0.001. These cases are reached when $Nf = \{10, 25\}$ for the HME algorithm and $Nf = \{400, 600\}$ for HSS.

Table 6: The average gap criteria when the number of files changes for all algorithms

Nf	LSF	HGC	SOH	HGS	HME	HSS
10	0.094	0.079	0.079	0.079	**0.000**	0.076
25	0.127	0.121	0.281	0.072	**0.000**	0.072
40	0.122	0.118	0.467	0.081	0.025	0.030
50	0.213	0.187	0.313	0.088	0.037	0.033
150	0.208	0.201	0.423	0.053	0.124	0.023
250	0.148	0.133	0.355	0.097	0.037	0.068
350	0.393	0.361	0.374	0.122	0.115	0.074
400	0.049	0.049	0.532	0.040	0.009	**0.000**
500	0.103	0.100	0.567	0.086	0.003	0.002
600	0.062	0.062	0.533	0.042	0.016	**0.000**

Table 7 shows the average gap criteria when the number of virtual machines changes for all algorithms. There are only two cases when the average gap is less than 0.001. These cases are reached when $Nv = \{5, 10\}$ for HSS. The advantage to execute HME is to reach a minimum average gap for almost Nf values. The execution time of HME is polynomial.

Table 7: The average gap criteria when the number of virtual machines changes for all algorithms

Nv	LSF	HGC	SOH	HGS	HME	HSS
3	0.263	0.223	0.229	0.123	0.016	0.097
4	0.243	0.227	0.217	0.097	0.123	0.092
5	0.090	0.090	0.490	0.053	0.020	**0.000**
10	0.123	0.122	0.512	0.057	0.029	**0.000**

Table 8 presents the average gap criteria when the classes change for all algorithms. This table shows that for *LSF*, *HGC*, *SOH*, and *HGS* the highest average gaps are observed for classes 3 and 4. This means that these classes are harder than classes 1 and 2 for these algorithms. However, for *HME*, the highest average gaps are observed for classes 1 and 2. This means that these classes are harder than classes 3 and 4 for this algorithm. Finally, for *HSS*, the highest average gaps are observed for classes 2 and 4. This means that these classes are harder than classes 1 and 3 for this algorithm.

Table 8: The average gap criteria when the classes change for all algorithms

Class	*LSF*	*HGC*	*SOH*	*HGS*	*HME*	*HSS*
1	0.154	0.151	0.524	0.037	0.061	0.024
2	0.166	0.153	0.603	0.076	0.044	0.065
3	0.188	0.171	0.181	0.104	0.035	0.038
4	0.186	0.163	0.197	0.111	0.027	0.045

Fig. 4 shows the average gap variation for *SOH* and *HGS* when the pair (*Nf*, *Nv*) changes. So, for each value (*Nf*, *Nv*), a pair value is given and presented in the figure with the related average gap. This figure shows that the curve of *HGS* is always below the curve of *SOH*. Indeed, the average gaps obtained by *SOH* are better than those obtained by *HGS*.

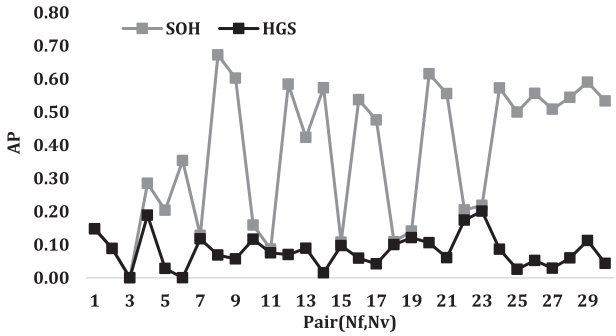

Figure 4: The average gap variation for ***SOH*** and ***HGS***

Fig. 5 shows the average gap variation for *HME* and *HSS* when the pair (*Nf*, *Nv*) changes. This figure shows that the curve of *HSS* is 16 times below the curve of *HME* and 15 times the opposite.

Despite the performance of the proposed algorithms, a hard instance can be generated with big-scale ones. In addition, the studied problem does not take into consideration when the virtual machines do not have the same characteristics. Indeed, in the studied problem, we suppose that all virtual machines have the same characteristics.

Table 9 represents results comparisons with other state of the art studies.

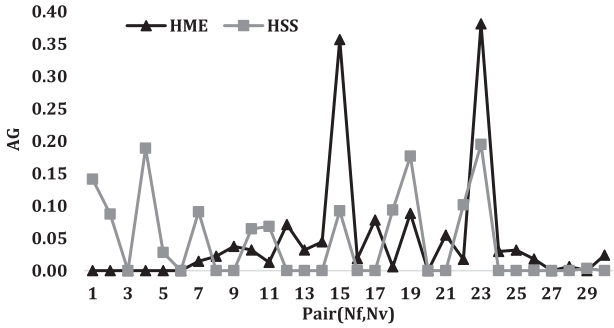

Figure 5: The average gap variation for ***HME*** and ***HSS***

Table 9: Results comparisons with other state of the art studies

Reference	Year	Applied algorithms	Treated problem	Scheduling strategy	Results
[55]	2022	ACO	Load balancing	Simulation	Load balancing and task scheduling strategies in the cloud environment
[56]	2021	Meta-heuristics methods	Cost and equity	Multi-objective grey wolf optimizer	Enhance cloud manufacturing (CMfg) scheduling
[57]	2022	Hybrid swarm intelligence method	Makespan time and cloud throughput	Math analyzes	Handle the problem of scheduling IoT tasks in cloud computing
[58]	2023	BCSV scheduling algorithm	Load balance and makespan	Scenarios	Load balancing problem of working nodes
Our paper	2022	Heuristics	Load balance and security awareness	Iterative, grouping	The used spaces in the cloud will be load balanced

6 Conclusion

In this paper, a new approach is proposed to schedule workflow in the cloud environment with utmost trust. Developed algorithms enable the scheduling process to choose the virtual machines that ensure load balancing. These algorithms provide more security to transferring workflow and minimize the time of data recovery in case of data loss. Our model is composed of three agents: the collector,

scheduler, and monitor. This model permits to visualization of the network links and node states permanently. The developed algorithms in the scheduler agent show promising results in terms of time and data protection. These algorithms are based on the grouping of several files into two groups. The choice of the file that is scheduled on the appropriate virtual machine is the most advantage of the work. Several iterative procedures are used in this paper. An experimental result is discussed using different metrics to show the performance of the proposed algorithms. In addition, four classes of instances are generated and tested. In total, there are 1240 instances in the experimental result. The experimental results show that the best algorithm is *HME*, with a percentage of 91.3%, an average gap of 0.042, and a running time of 0.001 s. The first line for future work is to enhance the proposed algorithms by applying several metaheuristics and considering the proposed algorithms as the initial solution. The second line is to propose a lower bound for the studied problem. The third line is to develop an exact solution and the last line is to develop intelligent algorithms for the monitor agent. After selecting the best algorithm, future research will focus on collector and monitor agents. The development of effective collection and monitoring agents enables the collection and analysis of meaningful information about virtual machines and intermediaries to decide on trusted bindings.

Acknowledgement: The authors extend their appreciation to the deputyship for Research & Innovation, Ministry of Education in Saudi Arabia.

Funding Statement: The authors extend their appreciation to the deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the Project Number (IFP-2022-34).

Author Contributions: Study conception and design: Mahdi Jemmali, Sarah Mustafa Eljack, Mohsen Denden; data collection: Mutasim Al Sadig; analysis and interpretation of results: Abdullah M. Algashami, Mahdi Jemmali, Sadok Turki, Mohsen Denden; draft manuscript preparation: Mahdi Jemmali, Sarah Mustafa Eljack. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data underlying this article are available in the article. All used materials are detailed in the experimental results section.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] H. Singh, S. Tyagi, P. Kumar, S. S. Gill and R. Buyya, "Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions," *Simulation Modelling Practice and Theory*, vol. 111, pp. 102353, 2021.

[2] X. Huang, C. Li, H. Chen and D. An, "Task scheduling in cloud computing using particle swarm optimization with time-varying inertia weight strategies," *Cluster Computing*, vol. 23, no. 2, pp. 1137–1147, 2020.

[3] A. Choudhary, I. Gupta, V. Singh and P. K. Jana, "A GSA-based hybrid algorithm for bi-objective workflow scheduling in cloud computing," *Future Generation Computer Systems*, vol. 83, pp. 14–26, 2018.

[4] S. Raghavan, P. Sarwesh, C. Marimuthu and K. Chandrasekaran, "Bat algorithm for scheduling workflow applications in cloud," in *2015 Int. Conf. on Electronic Design, Computer Networks & Automated Verification (EDCAV)*, Meghyala, India, IEEE, pp. 139–144, 2015.

[5] E. N. Alkhanak, S. P. Lee and S. U. R. Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities," *Future Generation Computer Systems*, vol. 50, pp. 3–21, 2015.
[6] M. Jemmali, "An optimal solution for the budgets assignment problem," *RAIRO-Operations Research*, vol. 55, no. 2, pp. 873–897, 2021.
[7] M. Alharbi and M. Jemmali, "Algorithms for investment project distribution on regions," *Computational Intelligence and Neuroscience*, vol. 2020, pp. 1–13, 2020.
[8] M. Jemmali, "Budgets balancing algorithms for the projects assignment," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 11, pp. 574–578, 2019.
[9] M. Jemmali, "Projects distribution algorithms for regional development," *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 10, no. 3, pp. 293–305, 2021.
[10] H. Alquhayz, M. Jemmali and M. M. Otoom, "Dispatching-rule variants algorithms for used spaces of storage supports," *Discrete Dynamics in Nature and Society*, vol. 2020, pp. 1–15, 2020.
[11] M. Jemmali, L. K. B. Melhim, M. T. Alharbi, A. Bajahzar and M. N. Omri, "Smart-parking management algorithms in smart city," *Scientific Reports*, vol. 12, no. 1, pp. 1–15, 2022.
[12] M. Jemmali and H. Alquhayz, "Equity data distribution algorithms on identical routers," in *Int. Conf. on Innovative Computing and Communications*, New Delhi, India, Springer, pp. 297–305, 2020.
[13] H. Alquhayz and M. Jemmali, "Max-min processors scheduling," *Information Technology and Control*, vol. 50, no. 1, pp. 5–12, 2021.
[14] S. A. Alsaidy, A. D. Abbood and M. A. Sahib, "Heuristic initialization of PSO task scheduling algorithm in cloud computing," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 2370–2382, 2022.
[15] D. A. Alboaneen, H. Tianfield and Y. Zhang, "Glowworm swarm optimisation based task scheduling for cloud computing," in *Proc. of the Second Int. Conf. on Internet of things, Data and Cloud Computing*, New York, USA, pp. 1–7, 2017.
[16] P. Durgadevi and D. S. Srinivasan, "Task scheduling using amalgamation of metaheuristics swarm optimization algorithm and cuckoo search in cloud computing environment," *Journal for Research*, vol. 1, no. 9, pp. 10–17, 2015.
[17] A. Belgacem, K. Beghdad-Bey and H. Nacer, "Task scheduling optimization in cloud based on electromagnetism metaheuristic algorithm," in *2018 3rd Int. Conf. on Pattern Analysis and Intelligent Systems (PAIS)*, Tebessa Algeria, IEEE, pp. 1–7, 2018.
[18] R. Masadeh, N. Alsharman, A. Sharieh, B. A. Mahafzah and A. Abdulrahman, "Task scheduling on cloud computing based on sea lion optimization algorithm," *International Journal of Web Information Systems*, vol. 17, no. 2, pp. 99–116, 2021.
[19] M. Abdullahi, M. A. Ngadi, S. I. Dishing and S. I. M. Abdulhamid, "An adaptive symbiotic organisms search for constrained task scheduling in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, pp. 8839–8850, 2023.
[20] I. Strumberger, N. Bacanin, M. Tuba and E. Tuba, "Resource scheduling in cloud computing based on a hybridized whale optimization algorithm," *Applied Sciences*, vol. 9, no. 22, pp. 4893, 2019.
[21] N. Bacanin, E. Tuba, T. Bezdan, I. Strumberger and M. Tuba, "Artificial flora optimization algorithm for task scheduling in cloud computing environment," in *Int. Conf. on Intelligent Data Engineering and Automated Learning*, Manchester, UK, Springer, pp. 437–445, 2019.
[22] N. Mansouri, B. M. H. Zade and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Computers & Industrial Engineering*, vol. 130, pp. 597–633, 2019.
[23] J. Ge, Q. He and Y. Fang, "Cloud computing task scheduling strategy based on improved differential evolution algorithm," *AIP Conference Proceedings*, vol. 1834, no. 1, pp. 40038, 2017.
[24] Y. Li, S. Wang, X. Hong and Y. Li, "Multi-objective task scheduling optimization in cloud computing based on genetic algorithm and differential evolution algorithm," in *37th Chinese Control Conf. (CCC)*, Wuhan, China, IEEE, pp. 4489–4494, 2018.

[25] Z. Zhou, F. Li and S. Yang, "A novel resource optimization algorithm based on clustering and improved differential evolution strategy under a cloud environment," *Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 5, pp. 1–15, 2021.
[26] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 4, pp. 4313–4327, 2023.
[27] J. Chen, P. Han, Y. Liu and X. Du, "Scheduling independent tasks in cloud environment based on modified differential evolution," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 13, pp. e6256, 2023.
[28] M. AbdElaziz, S. Xiong, K. Jayasena and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Systems*, vol. 169, pp. 39–52, 2019.
[29] X. Shi, X. Zhang and M. Xu, "A self-adaptive preferred learning differential evolution algorithm for task scheduling in cloud computing," in *IEEE Int. Conf. on Advances in Electrical Engineering and Computer Applications (AEECA)*, Dalian, China, IEEE, pp. 145–148, 2020.
[30] M. Abdel-Basset, R. Mohamed, W. AbdElkhalik, M. Sharawi and K. M. Sallam, "Task scheduling approach in cloud computing environment using hybrid differential evolution," *Mathematics*, vol. 10, no. 21, pp. 4049, 2022.
[31] X. Su, L. An, Z. Cheng and Weng, "Cloud-edge collaboration-based bi-level optimal scheduling for intelligent healthcare systems," *Future Generation Computer Systems*, vol. 141, pp. 28–39, 2023.
[32] M. Driss, A. Aljehani, W. Boulila, H. Ghandorh and M. Al-Sarem, "Servicing your requirements: An FCA and RCA-driven approach for semantic web services composition," *IEEE Access*, vol. 8, pp. 59326–59339, 2020.
[33] F. A. Ghaleb, M. A. Maarof, A. Zainal, B. A. S. Al-rimy, A. Alsaeediet *et al.,* "Ensemble-based hybrid context-aware misbehavior detection model for vehicular ad hoc network," *Remote Sensing*, vol. 11, no. 23, pp. 2852, 2019.
[34] W. Boulila, Z. Ayadi and I. R. Farah, "Sensitivity analysis approach to model epistemic and aleatory imperfection: Application to land cover change prediction model," *Journal of Computational Science*, vol. 23, pp. 58–70, 2017.
[35] M. Al-Sarem, F. Saeed, A. Alsaeedi, W. Boulila and T. Al-Hadhrami, "Ensemble methods for instance-based arabic language authorship attribution," *IEEE Access*, vol. 8, pp. 17331–17345, 2020.
[36] M. A. Al-Shareeda, S. Manickam and M. A. Saare, "DDoS attacks detection using machine learning and deep learning techniques: Analysis and comparison," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 2, pp. 930–939, 2023.
[37] M. A. Al-Shareeda, S. Manickam, S. A. Laghari and A. Jaisan, "Replay-attack detection and prevention mechanism in Industry 4.0 landscape for secure SECS/GEM communications," *Sustainability*, vol. 14, no. 23, pp. 15900, 2022.
[38] M. A. Al-Shareeda and S. Manickam, "COVID-19 vehicle based on an efficient mutual authentication scheme for 5G-enabled vehicular fog computing," *International Journal of Environmental Research and Public Health*, vol. 19, no. 23, pp. 15618, 2022.
[39] M. A. Al-Shareeda and S. Manickam, "MSR-DoS: Modular square root-based scheme to resist denial of service (DoS) attacks in 5G-enabled vehicular networks," *IEEE Access*, vol. 10, pp. 120606–120615, 2022.
[40] Y. S. Abdulsalam and M. Hedabou, "Security and privacy in cloud computing: Technical review," *Future Internet*, vol. 14, no. 1, pp. 11, 2022.
[41] M. Haouari, A. Gharbi and M. Jemmali, "Bounding strategies for scheduling on identical parallel machines," in *Int. Conf. on Service Systems and Service Management*, Troyes, France, IEEE, vol. 2, pp. 1162–1166, 2006.
[42] F. Al Fayez, L. K. B. Melhim and M. Jemmali, "Heuristics to optimize the reading of railway sensors data," in *6th Int. Conf. on Control, Decision and Information Technologies (CoDIT)*, Paris, France, IEEE, pp. 1676–1681, 2019.

[43] L. Hidri and M. Jemmali, "Near-optimal solutions and tight lower bounds for the parallel machines scheduling problem with learning effect," *RAIRO-Operations Research*, vol. 54, no. 2, pp. 507–527, 2020.
[44] A. B. Hmida and M. Jemmali, "Near-optimal solutions for mold constraints on two parallel machines," *Studies in Informatics and Control*, vol. 31, no. 1, pp. 71–78, 2022.
[45] O. A. Wahab, J. Bentahar, H. Otrok and A. Mourad, "Optimal load distribution for the detection of VM-based DDoS attacks in the cloud," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 114–129, 2017.
[46] G. Rjoub, J. Bentahar, O. A. Wahab, R. Mizouni, A. Song *et al.,* "A survey on explainable artificial intelligence for network cybersecurity," arXiv preprint arXiv:2303.12942, 2023.
[47] S. Arisdakessian, O. A. Wahab, A. Mourad and H. Otrok, "Towards instant clustering approach for federated learning client selection," in *Int. Conf. on Computing, Networking and Communications (ICNC)*, Honolulu, HI, USA, IEEE, pp. 409–413, 2023.
[48] G. Vijayakumar and R. K. Bharathi, "Streaming big data with open-source: A comparative study and architectural recommendations," in *Int. Conf. on Sustainable Computing and Data Communication Systems (ICSCDS)*, Erode, India, pp. 1420–1425, 2023.
[49] A. A. Fairosebanu and A. C. N. Jebaseeli, "Data security in cloud environment using cryptographic mechanism," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 1, pp. 462–471, 2023.
[50] M. Jemmali, M. Denden, W. Boulila, R. H. Jhaveri, G. Srivastava *et al.,* "A novel model based on window-pass preferences for data-emergency-aware scheduling in computer networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 7880 –7888, 2022.
[51] O. A. Wahab, J. Bentahar, H. Otrok and A. Mourad, "Resource-aware detection and defense system against multi-type attacks in the cloud: Repeated bayesian stackelberg game," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 605–622, 2019.
[52] M. Skafi, M. M. Yunis and A. Zekri, "Factors influencing smes' adoption of cloud computing services in lebanon: An empirical analysis using toe and contextual theory," *IEEE Access*, vol. 8, pp. 79169–79181, 2020.
[53] B. B. Gupta, K. C. Li, V. C. Leung, K. E. Psannis, S. Yamaguchi *et al.,* "Blockchain-assisted secure fine-grained searchable encryption for a cloud-based healthcare cyber-physical system," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 12, pp. 1877–1890, 2021.
[54] M. Jemmali, L. K. B. Melhim and F. Al Fayez, "Real-time read-frequency optimization for railway monitoring system," *RAIRO-Operations Research*, vol. 56, no. 4, pp. 2721–2749, 2022.
[55] Y. Natarajan, S. Kannan and G. Dhiman, "Task scheduling in cloud using ACO," *Recent Advances in Computer Science and Communications*, vol. 15, no. 3, pp. 348–353, 2022.
[56] B. Vahedi-Nouri, R. Tavakkoli-Moghaddam, Z. Hanzálek, H. Arbabi and M. Rohaninejad, "Incorporating order acceptance, pricing, and equity considerations in the scheduling of cloud manufacturing systems: Metaheuristics methods," *International Journal of Production Research*, vol. 59, no. 7, pp. 2009–2027, 2021.
[57] I. Attiya, M. Abd Elaziz, L. Abualigah, T. N. Nguyen and A. A. Abd El-Latif, "An improved hybrid swarm intelligence for scheduling IoT application tasks in the cloud," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6264–6272, 2022.
[58] M. L. Chiang, H. C. Hsieh, Y. H. Cheng, W. L. Lin and B. H. Zeng, "Improvement of tasks scheduling algorithm based on load balancing candidate method under cloud computing environment," *Expert Systems with Applications*, vol. 212, pp. 118714, 2023.