IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 3 July 2023, accepted 17 July 2023, date of publication 26 July 2023, date of current version 30 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3299213

RESEARCH ARTICLE

Efficient Storage Approach for Big Data Analytics: An Iterative-Probabilistic Method for Dynamic Resource Allocation of Big Satellite Images

MAHDI JEMMALI[1,2,3], WADII BOULILA[4,5], ASMA CHERIF[6,7], AND MAHA DRISS[5,8]

[1]MARS Laboratory, University of Sousse, Sousse 4002, Tunisia
[2]Department of Computer Science, Higher Institute of Computer Science and Mathematics of Monastir, University of Monastir, Monastir 5000, Tunisia
[3]College of Computing and Informatics, University of Sharjah, Sharjah, United Arab Emirates
[4]Robotics and Internet-of-Things Laboratory, Prince Sultan University, Riyadh 12435, Saudi Arabia
[5]RIADI Laboratory, National School of Computer Sciences, University of Manouba, Manouba 2010, Tunisia
[6]Information Technology Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[7]Center of Excellence in Smart Environment Research, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[8]Computer Science Department, College of Computer and Information Sciences, Prince Sultan University, Riyadh 12435, Saudi Arabia

Corresponding author: Mahdi Jemmali (mah_jem_2004@yahoo.fr)

This research work was funded by Institutional Fund Projects under grant no. (FPIP:1412-612-1443). The authors gratefully acknowledge technical and financial support provided by the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

ABSTRACT Satellite images play a crucial role in ecology as they provide rich information about the Earth's surface. The deep analysis of satellite images presents a vast challenge due to the sheer size of the data that needs to be managed. Sophisticated storage solutions are required to handle the ever-increasing velocity of incoming data and to deal with potential latency or data loss. Storage balancing ensures efficient allocation and distribution of storage capacity across a system, which involves monitoring, analyzing, and adjusting how data is stored to optimize performance, minimize downtime, and maximize cost savings. Additionally, storage balancing helps avoid data bottlenecks by automatically redistributing data across multiple resources. While many solutions have been proposed to balance storage, no polynomial solution is available. This paper addresses the issue of transmitting a considerable amount of satellite images across the network to various storage supports. The challenge is to find an effective way to schedule these satellite images to the storage supports that lead to equitable results in distribution. Many heuristics and enhancement methods are proposed to solve this problem. The effectiveness of the algorithms presented in this paper was tested and analyzed through extensive testing. The experimental study shows that the proposed heuristics outperform those developed in the literature. Indeed, in 73.8% of cases, the best-proposed algorithm, the best iterative-selection satellite images algorithm (*BIS*), reached the best solution compared to the best algorithm in the literature and the other proposed algorithms. The *BIS* algorithm obtained an average gap of 0.147 in an average running time of 1.0654 s.

INDEX TERMS Satellite images, big data, storage load balancing, heuristics, algorithms, approximate solutions.

I. INTRODUCTION

Satellite images have become a valuable tool in ecological research, providing researchers with a way to gather data on large-scale patterns and processes in the natural world. By analyzing satellite images, researchers can study the distribution and abundance of species [1], track changes in land use and land cover [2], [3], [4], biodiversity and forest disease monitoring [1], and disaster management [5]. This has led to new insights into how natural systems work and how they respond to human impacts, which can inform conservation efforts and help us better understand the complex relationships between different components of the environment [6]. Thus, satellite images provide a wide range of data that can be used to understand better and protect the Earth's ecosystems.

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Kashif Bashir.

Nonetheless, storing satellite images can be difficult for many reasons. Firstly, satellite images are typically high resolution and therefore require a large amount of storage space. This can quickly become an issue for organizations that need to store a large number of images or who need to store images for long periods. Additionally, organizing and managing large amounts of satellite image data can be complex, especially when dealing with multiple satellites, sources, and formats. These factors can make satellite image storage a challenging and resource-intensive task.

Load balancing is necessary to ensure that satellite image data is stored efficiently and optimally across multiple storage systems so that it can be accessed quickly and easily. By load-balancing storage, the system can better manage large volumes of data, allowing for faster access times and improved scalability. This is especially important in big data environments where large amounts of data need to be processed quickly.

The astonishing expansion of networks for data transmission, coupled with the tremendous demand for storage space by users in the Big Data context, is compelling computer experts to seek a solution to manage storage more effectively. Proper management leads to efficient resource allocation. Several algorithms can be devised to bridge the disparity in free space across different storage media.

Randomization methods employ an initial load-balancing technique that randomly assigns satellite images to the storage systems. However, randomly dispatching these images across storage media can result in uneven storage space usage. This imbalance may lead to inefficient resource distribution, where some storage media may end up completely filled while others remain half-empty or even empty. Iterative solutions have been proposed to consider system parameters such as satellite image size, type, and resource availability to ensure that the load on each storage system is balanced while considering the system constraints and parameters.

This paper proposes five heuristic-based algorithms to find the best way to store satellite images in the storage support, ensuring equitable storage. Firstly, we propose three enhancement phases be included in the proposed algorithms. These phases are based essentially on the probabilistic method. Indeed, we propose a method to apply the probability of the algorithm.

Our choice to use heuristics rather than artificial intelligence (AI) models is based on the following benefits provided by heuristic-based algorithms:

1) Interpretability: Heuristics are typically more straightforward and easier to understand than AI models, which can be complex and opaque. Heuristic algorithms often involve a series of logical rules that humans can understand, whereas AI models may involve complex mathematical functions that are difficult to interpret.
2) Efficiency: Heuristic algorithms can be faster and require less computational power than AI models, especially when dealing with small datasets or problems with simple structures. In some cases, a heuristic algorithm may be able to solve a problem in real-time, while an AI model may require significant processing time.
3) Robustness: Heuristics are often more robust than AI models in situations where the data is noisy or incomplete. Heuristic algorithms can often handle missing or inconsistent data and can make reasonable decisions based on partial information. In contrast, AI models can be sensitive to small changes in the input data and may produce inaccurate results if the data is noisy or incomplete.
4) Domain-specific knowledge: Heuristics can incorporate domain-specific knowledge and expertise, which can be difficult to capture in an AI model. For example, in a scheduling optimization problem, a heuristic algorithm may use a set of rules based on scheduling knowledge and experience to identify potential solutions, while an AI model would need to learn these rules from a large dataset.

A comparative experimental study was conducted to evaluate the effectiveness of the proposed algorithms against the best solutions identified in the literature review.

The main contributions of this paper are summarized in the following points:

- The development and implementation of five novel algorithms for solving the problem of satellite image allocation for storing large satellite images.
- The proposal of an innovative approach utilizing iterative-probabilistic techniques, probability distribution, iterative testing, and diverse satellite image choices, leading to an improved storage solution.
- The experimental testing has demonstrated the superiority of the proposed algorithms over existing heuristics. The best algorithm achieved a success rate of 73.87%, significantly surpassing the previous method's 63.1% success rate.
- An efficient running time of the algorithms without compromising accuracy, enabling more iterations and improved results' performance.
- The utilization of the proposed algorithms as effective initial solutions for various metaheuristics has facilitated faster convergence and better optimization results.
- There is no dominance among the proposed algorithms, offering researchers and practitioners multiple optimization options. Combining two or more algorithms can potentially enhance performance and provide different trade-offs between running time and accuracy.

This paper is organized as follows. Section II presents a state-of-the-art review of the problem under study. Section III provides a detailed problem description, while Section IV outlines the developed heuristics with three enhancement phases. In Section V, an experimental study was conducted to compare the performance of the developed heuristics against the best one identified in the literature. Finally, Section VI presents the conclusions of this study.

II. RELATED WORKS

Load-balancing algorithms have been the subject of numerous research works across various fields, including cloud computing and networking.

Several studies have investigated distributed load-balancing techniques in the context of cloud computing. Notably, [7], [8], [9], [10], [11] have all conducted comparative analyses of such techniques.

In [12], authors examined the problem of load balancing in distributed file systems. They also analyzed the impact of different factors on load balancing. They developed a load-balancing algorithm in the presence of multiple servers, aiming to sort the most and least loaded servers. An additional study proposed in [13] has proposed a heuristic algorithm to optimize the assignment of tasks to virtual machines (VMs) in cloud computing environments. The approach optimizes the usage of virtual machines (VMs) more effectively. It is based on a fair tasks distribution method utilized in the infrastructure as a service (IaaS) model based on the number of incoming tasks and their sizes while maximizing the utilization of computing resources. Furthermore, [14] developed a load-balancing algorithm to enhance cloud performance and conducted a comparative analysis of several load-balancing algorithms in cloud computing. The study proposed an improved load-balancing architecture that divides the functions of the main controller into two parts. The first part involves partitioning user tasks through regional load balancers, while the second part ensures system updating through various agents. The Cloud architecture is organized into three levels, each with specific roles and algorithms. The aim is to optimize the response times of Cloud services by selecting the closest load balancer based on the geographical location of the user and data center and allocating appropriate nodes to user tasks using an ant colony optimization algorithm. The author in [15] presented a mathematical model of a dispatching problem and proposed several heuristics to distribute processors among manufacturing machines efficiently. They proved that the problem has an NP-hard complexity. The heuristics used in their approach included dispatching rules, a swapping method, and a mixture approach. The experimental results showed that the most effective heuristic was $SIDA^r$. In [16], the authors proposed a rapid algorithm that utilizes the zero imbalance strategy. This strategy aims to minimize the completion time gap among heterogeneous VMs without needing priority approaches or complex assignment decisions for the cloud configuration. The authors defined two constraints: the earliest and optimal completion time. Their proposed algorithm is able to solve the NP-hard optimization problem while satisfying both the users' and providers' constraints. Gupta proposed in [17] two distributed load-balancing algorithms, CDLB and DDLB, that leverage multiple cloud storage server parameters to optimize performance. The first algorithm mainly considers the service rate and queue length, while the second takes into account additional parameters such as service time and deadline time for client requests. Their work monitors various factors that improve the overall performance of cloud storage. Both algorithms attempt to evenly distribute the load across storage servers while fully utilizing server capabilities. Simulation results demonstrate that these proposed algorithms effectively balance the load, maximize server utilization, shorten response times, and improve system performance. n the context of high-performance computing (HPC), [18] proposed an enhancement to the max-min scheduling method. This method prioritizes tasks with the maximum execution time before assigning those with the minimum execution time, which reduces the delay in executing tasks with a lower execution time. The authors utilized supervised machine learning to delegate the most significant cluster requests to the virtual machine with the least usage when clusters use the size of clusters and a portion of virtual machines.

Alqarni et al. [19] introduced a load-balancing framework based on the Binary Cuckoo Search Algorithm to decide whether tasks should be executed locally or at edge nodes. The framework is designed to minimize time, energy, and payment costs associated with offloading tasks by formulating the problem as a mixed-integer optimization problem.

Load-balancing algorithms have also found applications in networking. Karger et al. [20] proposed two simple and efficient load-balancing algorithms for peer-to-peer systems. The first algorithm ensures that the key address space is distributed in a balanced way among nodes, such that the random mapping of items achieves load balancing. The second algorithm directly balances the distribution of items among nodes. The study demonstrates the effectiveness of both algorithms in improving load balancing in peer-to-peer systems. However, these algorithms have some limitations; for instance, the model is not easily generalizable to more than one order.

Jemmali et al. [21] tackled the problem of distributing large data packets across different routers while ensuring equal sending times. The researchers proposed a novel network architecture with a scheduler component that utilizes multiple algorithms to minimize the time gap between sending data packets. They developed four heuristics and conducted experiments to compare their performance. The results demonstrated that the proposed heuristics performed effectively.

An adaptive mobility load-balancing algorithm was developed in [22] to balance the load across LTE small-cell networks by utilizing network load status. The algorithm uses a threshold-based approach to determine when and where to hand over users between base stations. The threshold values are determined based on the network load and user mobility. The proposed algorithm was tested through simulations, and the results showed that it effectively reduced the number of handovers and improved network performance. A patent was registered regarding the methods and systems

for load balancing in a cluster storage system [23]. The patent addresses the problem of balancing a load of data objects in different cluster nodes to optimize performance by transferring the data objects to other clusters based on threshold and proximity. Another patent was also registered for routing requests for information. The latter dealt with systems and methods for load balancing using predictive routing.

Several recent works treating load balancing applied to gas turbines were introduced in [24] and [25]. Load balancing can also be applied to other real-world scenarios, such as the distribution of projects across multiple regions in [26], [27], [28], and [29]. In addition, the maximum of the minimum completion time is treated as a solution to the parallel machines problem [30]. Indeed, in the aforementioned work, authors developed many heuristics to solve the load-balancing problem on machines. These heuristics were compared with state-of-the-art solutions. The experimental study showed the efficiency of the developed heuristics. Further, several other works that address load balancing are discussed in [31], [32], [33], and [34]. The real-time intervention system developed in [35] and [36] can be utilized to improve the proposed system.

Reference [37] proposed a system enabling dynamic data allocation across multiple nodes that were presented to balance the workload in a database. This system evaluates the utilization of various buffers as well as the servers' capacity to allocate data to their corresponding active server buffers. The system makes use of two types of buffers: one for accepting external data and another for storing data from active servers. Data is allocated to the active servers' buffers depending on their capacity factors and the buffer's current usage state. The quantity of data allotted is controlled by the buffer utilization ratio and the server's capability factor, with higher ratios corresponding to smaller data allocations and lower capability factors resulting in larger data allocations.

Table 1 provides a summary of the related works detailed previously.

These related works present several limitations that can be summarized as follows:

- Scalability: some load-balancing algorithms may not be easily scalable to accommodate the increasing number of files, servers, or nodes in a cloud computing or networking environment;
- Overhead: Some load-balancing algorithms may introduce additional overhead and computational costs, negatively impacting overall system performance;
- Lack of accuracy: some Load-balancing algorithms may not always accurately predict the load on servers or nodes, leading to inefficient load distribution and potentially reducing system performance;
- Single point of failure: Some load-balancing algorithms rely on a single centralized controller or node, which can become a single point of failure and compromise the reliability of the system;
- Limited applicability: Some load-balancing algorithms may only be applicable to certain types of files, systems, or architectures, limiting their overall usefulness.

To overcome these limitations, this paper presents five novel algorithms for efficiently storing large satellite images by distributing the workload across multiple servers. The algorithms are based on an iterative-probabilistic approach and have been tested extensively to demonstrate their superior performance compared to existing heuristics/algorithms. The proposed algorithms achieve high success rates and fast running times without sacrificing accuracy, making them an excellent initial solution for optimization processes. Furthermore, no dominant algorithm gives researchers and practitioners more options for optimizing their storage and load-balancing problems.

III. PROBLEM STATEMENT

In this section, we present the problem description and definition.

A. PROBLEM DESCRIPTION

In the context of managing satellite images, load balancing can play a critical role in ensuring that image processing tasks are distributed efficiently across the available resources, such as servers or computing clusters. Efficient load balancing is particularly important when managing large volumes of satellite images, which require significant processing power and storage resources. Many challenges can face load balancing for big satellite images, including:

- High computational requirements: Processing large satellite images can require significant computational resources, which can create a bottleneck in the load-balancing system. If the system is not designed to handle these high computational requirements, it can lead to slow processing times and delays in the delivery of results.
- Imbalanced workloads: Load balancing systems must be able to distribute workloads evenly across available resources. However, with big satellite images, there may be significant variations in image size, resolution, and processing requirements, which can result in imbalanced workloads. This can lead to some resources being overutilized while others are underutilized, reducing overall system efficiency.
- Data transfer limitations: Moving large satellite images between storage locations and processing resources can be time-consuming, especially if the system is not designed to handle high-speed data transfer. This can further slow down processing times and limit the effectiveness of the load-balancing system.
- Network latency: In distributed load balancing systems, network latency can create delays and impact system performance, especially when processing large satellite images. This can be exacerbated by factors such as network congestion or bandwidth limitations, further hindering load-balancing performance.

TABLE 1. Summary of related work.

Ref	Type	Purpose	Area	Methodology
[12]	Research	Distributed file system	Cloud computing	Fitness function and threshold
[13]	Research	Tasks distribution over VMs	IaaS	Heuristic & queuing
[14]	Research	Architectural design and swarm optimization	Cloud computing	Meta-heuristic (Ant colony)
[15]	Research	Processes distribution among manufacturing machines	Manufacturing	Heuristic
[16]	Research	Mathematical model	Cloud computing	Zero imbalance
[17]	Research	Mathematical model	Cloud computing	
[18]	Research		HPC	Machine learning
[20]	Research	Item distribution balancing in P2P networks	Networking	Key distribution and ordering
[19]	Research	Task distribution in edge computing	Edge computing	Meta-heuristic (Cuckoo search)
[21]	Research	Data packets distribution	Networking	Heuristic
[22]	Research	LTE small-cell networks	Networking	Threshold-based approach
[38]	Patent	Load balancing in cluster storage	Cloud computing	Threshold and proximity
[23]	Patent	Routing requests	Cloud Computing	Machine learning
[24], [25]	Research	Gas turbine maintenance	Gas turbines	Heuristic
[26]–[29]	Research	Distribution of projects across multiple regions	Scheduling	Heuristic
[30]	Research	Parallel machines problem	Parallel processing	Heuristic
[37]	Patent	Data allocation to servers	Database workload balancing	Dynamic allocation & thresholds

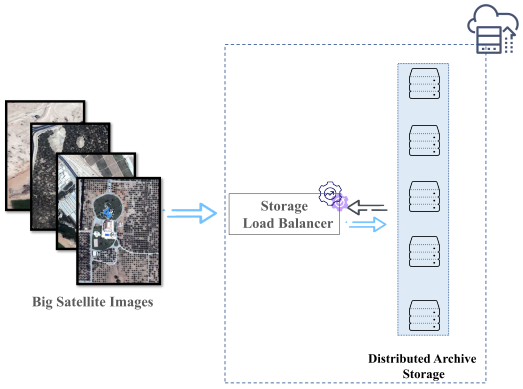


FIGURE 1. Model for the studied problem.

We propose a novel load-balancing method for big satellite images to address these challenges. The proposed approach will ensure better scalability, flexibility, and efficient resource utilization. This involves designing an efficient optimization-based technique to distribute a large amount of satellite data to enable effective load balancing and processing of large satellite image datasets. The proposed approach will ensure that the available resources are being utilized to their fullest potential, minimizing any downtime or delays that might otherwise occur.

The studied problem can be illustrated as detailed in Figure 1.

B. PROBLEM DEFINITION

In this paper, the studied problem can be presented as follows. Giving a set S of n_{sI} satellite images to be sent to a set of storage supports $ST = \{st_1, \cdots, st_{n_s}\}$. The number of storage supports is n_s. The size of each satellite image SI_j is s_j. Once the satellite image SI_j is stored, the cumulative size satellite image is denoted by cs_j. The total used space for the storage support st_i after the assignment of all satellite images is denoted by Us_i with $i = \{1, \cdots, n_s\}$.

The maximum (minimum) used space of the storage supports after the completion of the backup period is Us_{max} (Us_{min}). The objective is to reduce this gap. Thus, an appropriate schedule must be found for the related problem. The objective is to minimize the gap between the used space. This gap can be formulated and calculated by several methods. We choose in this paper the indicator that can calculate the gap as proposed in [15]. The total gap denoted by TC_g is written by Equation 1 below:

$$TC_g = \sum_{i=1}^{n_s} [Us_i - U_{min}] \tag{1}$$

The objective is to minimize TC_g. The studied problem will be denoted by $P \,||\, TC_g$. The problem is proved to be NP-hard in [15].

Proposition 1: The gap TC_g can be written as $\sum_{i=1}^{n_s} Us_i - n_s U_{min}$.

Proof: $TC_g = \sum_{i=1}^{n_s} [Us_i - U_{min}] = \sum_{i=1}^{n_s} Us_i - \sum_{i=1}^{n_s} U_{min}$. We have, $\sum_{i=1}^{n_s} U_{min} = n_s U_{min}$, so $\sum_{i=1}^{n_s} Us_i - n_s U_{min}$.

Example 1 explains the studied problem with a given instance.

Example 1: Suppose that $n_{SI} = 6$ and $n_s = 2$. The distribution of the size of each satellite image SI_j is given in Table 2.

TABLE 2. Satellite images size values for Example 1.

SI_j	SI_1	SI_2	SI_3	SI_4	SI_5	SI_6
s_j	5	6	3	12	4	14

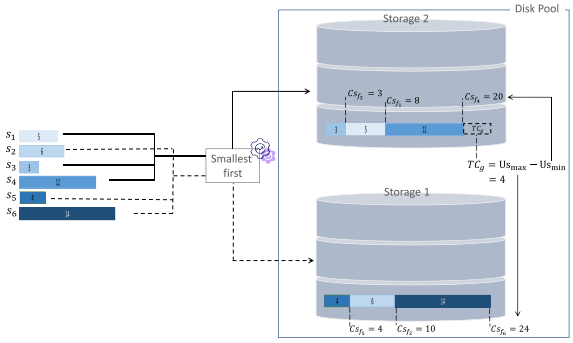


FIGURE 2. Schedule applying the algorithm of smallest satellite images first.

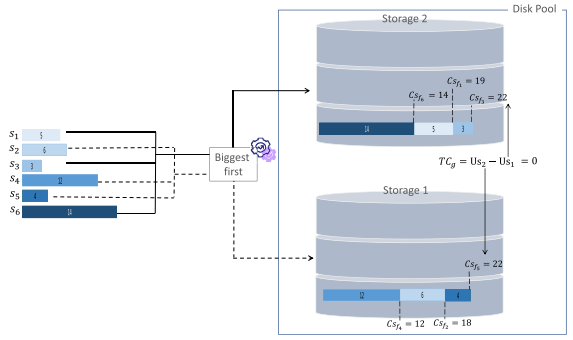


FIGURE 3. Schedule applying the algorithm of biggest satellite images first.

The objective is to find a schedule that stores all satellite images on the available storage supports. This can be accomplished by implementing the smallest satellite images first algorithm; the resulting schedule is shown in Figure 2.

The schedule depicted in Figure 2 indicates that the satellite images $\{SI_1, SI_3, SI_4\}$ are stored in the first storage support, while satellite images $\{SI_2, SI_5, SI_6\}$ are saved in storage support 2. Consequently, the used space in storage supports 1 and 2 is 20 and 24, respectively. This creates a gap between storage supports 1 and 2, equal to $TC_g = Us_2 - Us_1 = 4$.

Now, calling the biggest satellite images the first algorithm, the obtained schedule is illustrated in Figure 3.

The above schedule, depicted in Figure 2, shows that satellite images $\{SI_1, SI_3, SI_6\}$ are stored in the first storage supports, while satellite images $\{SI_2, SI_4, SI_5\}$ are saved in storage support 2, obtained by using the biggest satellite images first scheduling algorithm. This leads to used storage of 24 and 23 for storage supports 1 and 2, respectively. This results in a gap between the storage support 1 and 2, which is $TC_g = Us_2 - Us_1 = 0$. It is clear that Schedule 2 is better than Schedule 1 because TC_g is improved.

IV. BEST ALGORITHM IN THE LITERATURE

Our study aims to propose several heuristics to solve the problem under consideration. The proposed heuristics will be compared with those found in the literature review. The best heuristic proposed in the literature review will be presented, and an experimental study will be conducted to compare it with the proposed new ones.

In [15], the authors proposed several heuristics to solve the problem. A comparison was made to show the performance of different algorithms. Based on the assessment given in [15], the algorithm $SIDA^r$ was found to be the best in 75.2% of cases, with an average execution time of 0.115 s. This heuristic is described as the swapping non-increasing-decreasing sizes with order heuristic. It is based on swapping the satellite images, one by one, following the order specified by the swapping non-increasing-decreasing sizes algorithm $SIDA()$, but picking the biggest and smallest satellite images alternatively. To enhance the $SIDA$ algorithm, the choice of the biggest and smallest satellite images is repeated the number of times denoted by r, and the best solution is selected. $SIDA^r$ returns the best solution.

The algorithm for $SIDA^r$ as proposed in [15] is presented in Algorithm 1.

Algorithm 1 Swapping Non-Decreasing-Increasing Algorithm: $SIDA^r$

1: **for** $(t = 2$ to $t = r)$ **do**
2: Calculate $gap_t = SIDA_t(F)$
3: **end for**
4: Calculate $gap = \min_{2 \le t \le r} (gap_t)$
5: Return gap

V. NOVEL ENHANCEMENT PHASES

In this section, we propose three enhancement phases to improve the heuristics. We present and utilize these enhanced phases. The first phase is based on the randomization method, and the second phase is based on selecting the probability value through a chosen ratio. Finally, the third phase utilizes the iteration number of the initially proposed algorithm. This iterative method is an enhancement of any algorithm.

A. FIST ENHANCEMENT PHASE

The randomized heuristic is based on the following idea: selecting the first and second biggest satellite images. This means choosing the satellite image with the biggest size and the next biggest one. The probability β is then applied to decide between the two ordered satellite images. The first satellite image is selected with probability β and the next one with probability $1 - \beta$.

In practice, the probability is applied through a ratio denoted by r. This ratio takes on values of $r = 20, 30, 40$. We randomly generate a number n_r between 1 and 100. If $n_r < r$, we select the first biggest satellite image; otherwise, we select the second biggest. The value r can also

be looped between 5 and 50 with a step equal to 5. In this case, the value r can be presented as 5, 10, 15, . . . , 50.

B. SECOND ENHANCEMENT PHASE

In this phase, we adopt an iterative approach to enhance the value of the obtained heuristic. Instead of running the algorithm only once, we loop it multiple times, and at each iteration, a new probability value is returned, which leads to a new scheduled result. We denote by it the number of iterations that the algorithm can be looped. In this paper, we fix $it = 1$ for the execution of the algorithm once and $it = 100$ for the loop of running the algorithm 100 times. The best value among all iterations will be retained.

C. THIRD ENHANCEMENT PHASE

In this phase, we extend the choice between the first and second-biggest satellite images to include the third-biggest satellite images as well. Instead of selecting between the two biggest satellite images, we randomly choose between the three biggest satellite images using a uniform distribution.

We generate a random number n_r between 1 and 3 to implement this. If $n_r = 1$, we select the biggest satellite image; if $n_r = 2$, we select the second-biggest satellite image; if $n_r = 3$, we select the third-biggest satellite image.

By applying all three phases above, the resulting heuristic is denoted by $RH2^r_{it}$ when we choose between the two biggest satellite images and $RH3^r_{it}$ when we choose between all three biggest satellite images.

We use the function $Rand(X)$ to generate a random integer in the range $[1, X]$ and the procedure $Storing_B(a)$ to schedule the a biggest satellite images on the storage supports with the freest space.

VI. PROPOSED ALGORITHMS

In this section, we present twelve algorithms. The proposed algorithms are based on randomization and iterative method.

A. ITERATIVE SELECTED-RATIO VARIATION ALGORITHM (ISV)

Firstly, the selection of satellite images will be between the first biggest satellite image and the second one. The ratio r is fixed to 20 with no iteration for this algorithm. The randomization procedure is implemented in Algorithm 2, denoted by $RH2^{20}_1$.

Now, a modification of the ratio is applied to implement another procedure. The only modification is the choice of the ratio. Instead, to fix $r = 20$, we fix $r = 30$. If the procedure above runs with the first instruction change, $r = 20$ will be $r = 30$. A new result is obtained and denoted by $RH2^{30}_1$. The same idea is we change $r = 30$ to be $r = 40$, and a new result will be obtained and denoted by $RH2^{40}_1$. Therefore, the ratio variation algorithm denoted by RVA is defined as $SRV = \min(RH2^{20}_1, RH2^{30}_1, RH2^{40}_1)$.

Now the SRV will be run iteratively to obtain a new enhanced algorithm. We iterate the $RH2^{20}_1$ procedure 100 times as a first procedure. In each iteration, the returned solution is stored in an array. After finishing all iterations, the best solution is selected and returned among all ones stored in the latter arrays. We denote this best solution by $RH2^{20}_{100}$. We iterate the $RH2^{30}_1$ procedure 100 times as a second procedure. In each iteration, the returned solution is stored in an array. After finishing all iterations, the best solution is selected and returned among all ones stored in the latter arrays. Wed denotes this best solution by $RH2^{30}_{100}$. We iterate the $RH2^{40}_1$ procedure 100 times as a third procedure. In each iteration, the returned solution is stored in an array. After finishing all iterations, the best solution is selected and returned among all ones stored in the latter arrays. We denote this best solution by $RH2^{40}_{100}$. Therefore, the iterative-selected ratio variation algorithm (ISV) is defined as $ISV = \min(RH2^{20}_{100}, RH2^{30}_{100}, RH2^{40}_{100})$.

Algorithm 2 $RH2^{20}_1$ Procedure

```
Set r = 20, j = 1
while (j < n_f) do
    Determine n_r = Rand(100)
    if (n_r ≤ r) then
        Call Storage_B(1)
    else
        Call Storage_B(2)
    end if
    Set j++
end while
Save the last satellite image in the storage supports having the minimum used space
Calculate TC_g
Return TC_g
```

B. ITERATIVE LOOPING-RATIO VARIATION ALGORITHM (ILV)

This algorithm is based on a basic procedure called the looping ratio variation procedure and is denoted by LRV. A new perturbation of $RH2^{20}_1$ is applied to implement the new basic procedure of this algorithm. This perturbation is the choice of ratio. For this algorithm, we execute different values of the ratio. These values are based on the looping between $[5 - 50]$ of the ratio value with a fixed step of 5. Algorithm 3 presents the instructions for the looping ratio variation algorithm (LRV).

Now, we iterate the $RH2^{5-50}_1$ algorithm 100 times. In each iteration, the returned solution is stored in an array. After finishing all iterations, the best solution is selected and returned among all ones stored in the latter arrays. This algorithm is called the iterative looping-ratio variation algorithm and is denoted by ILV.

C. ITERATIVE THREE-BIGGEST-SELECTION SATELLITE IMAGES ALGORITHM (ITS)

This algorithm is based on the basic procedure called the three biggest satellite image selection procedures and is denoted

Algorithm 3 Looping Ratio Variation Procedure (*LRV*)

Set $r = 5$
while $(r \leq 50)$ **do**
  Set $j = 1$
  while $(j < n_f)$ **do**
    Set $n_r = Rand(100)$
    if $(n_r \leq r)$ **then**
      Call $Schedule_B(1)$
    else
      Call $Schedule_B(2)$
    end if
  end while
  Save the last satellite image in the storage supports having the minimum used space
  Calculate TC_g^r
  Set $r = r + 5$
end while
Calculate $TC_g = \min TC_g^r$
Return TC_g

by *TLS*. The procedure is described as follows. Firstly, the three biggest satellite images will be selected. A probability ω is then used to choose the first biggest satellite image. The second biggest satellite image is selected using a probability ϕ where $1 - \omega > \phi$. The third biggest satellite image is selected using the probability $1-\phi-\omega$. Algorithm 4 presents the instructions for the *TLS* algorithm.

Algorithm 4 Three Largest-Selection Satellite Images Procedure (*TLS*)

Set $j = 1$
while $(j < n_{SI})$ **do**
  Set $\theta \in \{0.1, \cdots, 1\}$
  if $(\theta \leq \omega)$ **then**
    Call $Storing_B(1)$
  else
    if $(\omega < \theta \leq \omega + \phi)$ **then**
      Call $Storing_B(2)$
    else
      Call $Storing_B(3)$
    end if
  end if
  Set $j++$
end while
Save the last satellite image in the storage supports having the minimum used space
Calculate TC_g
Return TC_g

Now, we iterate the *TLS* algorithm 100 times. In each iteration, the returned solution is stored in an array. After finishing all iterations, the best solution is selected and returned among all ones stored in the latter arrays. This algorithm is called the iterative three-largest-selection satellite images algorithm and is denoted by *ITS*.

D. ITERATIVE MINIMUM-THREE-VARIANTS-SELECTION SATELLITE IMAGES ALGORITHM (IMS)

We modify *TLS* to obtain the basic procedure of the presented algorithm. This modification is to change the number of the biggest satellite images three times. The first time we select the first biggest satellite images, we store them on the storage supports with the freest space, and so on, until we store all satellite images. We calculate the gap denoted by TC_g^1. The second time, we chose between the first biggest satellite images and the second one. We calculate the gap denoted by TC_g^2. The last time, we chose between the three biggest satellite images and calculated TC_g^3. After that, we calculate $TC_g = \min(TC_g^1, TC_g^2, TC_g^3)$. *MTS* denotes this basic procedure.

The next step of this algorithm is to iterate the *MTS* procedure 100 times. In each iteration, the returned solution is stored in an array. After finishing all iterations, the best solution is selected and returned among all ones stored in the arrays. This algorithm is called the iterative minimum-three-variants-selection satellite images algorithm and is denoted by *IMS*.

E. BEST ITERATIVE-SELECTION SATELLITE IMAGES ALGORITHM (BIS)

After extensive experimentation, it has been found that none of the proposed algorithms are dominant, meaning that there is no algorithm among them that can outperform all the remaining ones. Therefore, combining these algorithms to generate a new one based on the minimum value may lead to better results.

The first step in creating this new algorithm is to call the *IMS* algorithm and assign the obtained results to the variable TC_g^1. Then, the *ITS* algorithm is called, and the results are assigned to the variable TC_g^2. Finally, the *ILV* algorithm is called, and the results are assigned to the variable TC_g^3. The proposed new algorithm is called the best iterative-selection satellite images algorithm and is denoted by *BIS*. This algorithm calculates the minimum value between TC_g^k, where $k \in 1, 2, 3$. The returned value of *BIS* is denoted by TC_g, and thus $TC_g = \min(TC_g^1, TC_g^2, TC_g^3)$.

In this paper, five algorithms were proposed to solve the considered problem. The impact of parameter choice on the result was shown by using different parameters on each algorithm. It was found that the value returned by the algorithm can be impacted by choice of each parameter. This motivates the use of different values for the same parameter type. The results of each parameter value choice are detailed in the following section.

VII. EXPERIMENTATION

In this section, testing will be conducted on five classes of instances generated as detailed in [15].

TABLE 3. Satellite images-sizes and number of storage-supports distribution.

n_{SI}	n_s
10, 20, 25, 50	2, 3, 5
100, 150, 250, 300, 500	2, 3, 5, 10, 15, 20, 25, 30
1000, 1500, 2000, 2500, 3000, 3500	2, 3, 5, 10, 15, 20, 25, 30, 50

The C++ language was used to code the proposed algorithms. The heuristics were run on a computer with an Intel(R) Core (TM) i5-3337U CPU and Windows 10. The method for generating satellite image sizes s_j was implemented using two types of distributions. The satellite image size is given in gigabytes (GB). The tested classes are generated as follows:

- *Class A*: s_j is in $U[1, 100]$.
- *Class B*: s_j is in $U[10, 300]$.
- *Class C*: s_j is in $U[500, 1500]$.
- *Class D*: s_j is in $U[50, 250]$.
- *Class E*: s_j is in $U[25, 1000]$.

The uniforms and the normal distribution are denoted by U and N, respectively. The number of total instances depends on the selection of n_{SI}, n_s, and *Class*. The pair can obtain many permutations (n_{SI}, n_s). The values of (n_{SI}, n_s) are detailed in Table 3.

The performance of the proposed heuristics compared to the algorithm $SIDA^r$ developed in [15] will be assessed. A total of 5300 instances were used in the experimental results. The indicators that can assess the proposed algorithms are:

- H is the best algorithm value after the execution of all algorithms.
- H_d is the proposed algorithm.
- $G_H = \frac{H_d - H}{H_d}$, if $H_d = 0$ then $G_H = 0$.
- Gp is the average value of G_H over a fixed number of instances
- *Time* is the average execution time (seconds).
- *Per* is the percentage among 5300 instances when $H_d = H$.

Different analyses can be presented in this paper through several tables. Table 4 presents an overall review of all proposed heuristics, compared with the best one of the literature review, $SIDA^r$. This table presents the percentage among 5300 instances when $H_d = H$, the average gap for each algorithm for all instances, and the corresponding average time. It can be observed from Table 4 that the results obtained by the best-developed algorithm outperform those of the best algorithm from the literature. Indeed, for *BIS*, the percentage is 73.8% compared to 63.1% for $SIDA^r$. The average gap for *BIS* is 0.147, while the average gap for $SIDA^r$ is 0.197. The algorithm *ILV* is the second-best one among the developed algorithms, with a percentage of 69.1%, an average gap of 0.268, and an average running time of 0.8474 s. The algorithm *ISV* obtains the minimum *Per* value, which is equal to 47%, with an average gap of 0.268 and an average running time of 0.0761 s.

TABLE 4. Overall review of algorithms results.

	IMS	ITS	ILV	ISV	BIS	$SIDA^r$
Per	49.1%	48.6%	69.1%	47.0%	73.8%	63.1%
Gp	0.261	0.264	0.163	0.268	0.147	0.197
Time	0.1449	0.0731	0.8474	0.0761	1.0654	0.0938

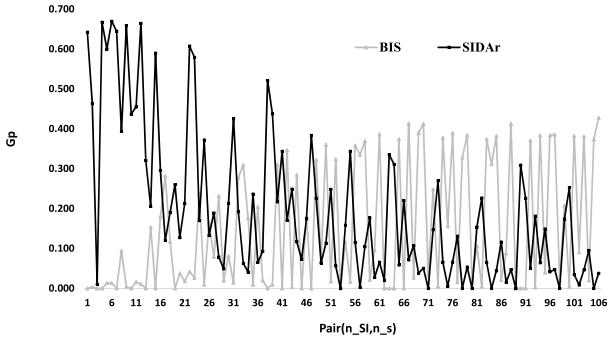


FIGURE 4. The *Gp* comparison between *BIS* and $SIDA^r$ when the pair (n_{SI}, n_s) changes.

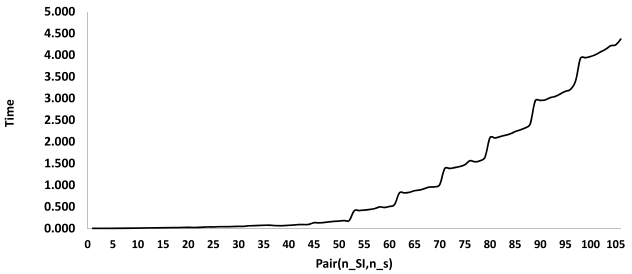


FIGURE 5. The *Time* behavior for *BIS* when the pair (n_{SI}, n_s) changes.

Table 5 presents the variations of *Gp* and *Time* for all algorithms when n_{SI} changes. Table 5 shows that the minimum average gap of 0.001 is achieved by *BIS* when $n_{SI} = 10$. For the algorithm $SIDA^r$, the minimum average gap of 0.072 is achieved when $n = 3500$. On the other hand, for the algorithm *BIS*, the maximum average gap of 0.215 is achieved when $n_{SI} = 2000$. For the algorithm $SIDA^r$, the maximum average gap of 0.646 is achieved when $n_{SI} = 20$. In comparison with *ILV*, the maximum gap of 0.217 is achieved when $n_{SI} = 3500$.

Table 6 presents the variation of *Gp* and *Time* for all algorithms when n_s changes. This table shows that the algorithm *BIS* reaches the minimum average gap of 0.007 when $n_s = 5$. However, the algorithm $SIDA^r$ obtains the minimum average gap of 0.050 when $n_s = 30$ and $m = 50$.

Figure 4 illustrates the comparison between *BIS* and $SIDA^r$ when the pair (n_{SI}, n_s) changes.

Table 7 presents the variation of *Gp* and *Time* for all algorithms when *Class* changes. We can observe from

TABLE 5. Variation of *Gp* and *Time* for all algorithms when n_{SI} changes.

n_{SI}	IMS		ITS		ILV		ISV		BIS		SIDA^r	
	Gp	Time	Gp	Time	Gp	Time	Gp	Time	Gp	Time	Gp	Time
10	0.084	0.0005	0.106	0.0005	0.070	0.0015	0.177	0.0002	0.001	0.0026	0.372	0.0000
20	0.316	0.0001	0.363	0.0003	0.051	0.0027	0.344	0.0001	0.009	0.0031	0.646	0.0000
25	0.311	0.0002	0.351	0.0004	0.137	0.0032	0.397	0.0002	0.033	0.0039	0.566	0.0000
50	0.346	0.0032	0.324	0.0007	0.047	0.0065	0.318	0.0006	0.009	0.0104	0.519	0.0001
100	0.243	0.0026	0.242	0.0018	0.112	0.0141	0.258	0.0025	0.096	0.0186	0.264	0.0006
150	0.300	0.0046	0.290	0.0027	0.113	0.0259	0.285	0.0027	0.101	0.0332	0.293	0.0011
250	0.241	0.0086	0.239	0.0050	0.155	0.0466	0.244	0.0046	0.137	0.0603	0.161	0.0022
300	0.300	0.0110	0.308	0.0063	0.132	0.0594	0.288	0.0057	0.122	0.0767	0.269	0.0030
500	0.272	0.0222	0.266	0.0138	0.164	0.1203	0.263	0.0116	0.159	0.1562	0.167	0.0076
1000	0.256	0.0659	0.254	0.0343	0.197	0.3635	0.255	0.0338	0.189	0.4636	0.111	0.0344
1500	0.251	0.1235	0.258	0.0628	0.192	0.7107	0.258	0.0648	0.180	0.8971	0.135	0.0741
2000	0.275	0.2016	0.282	0.1017	0.218	1.1815	0.279	0.1058	0.215	1.4849	0.082	0.1276
2500	0.233	0.3004	0.236	0.1497	0.204	1.7564	0.246	0.1657	0.191	2.2065	0.074	0.2019
3000	0.238	0.4173	0.235	0.2082	0.185	2.4626	0.245	0.2185	0.174	3.0881	0.119	0.2778
3500	0.268	0.5525	0.270	0.2776	0.217	3.2642	0.276	0.2837	0.212	4.0943	0.072	0.3764

TABLE 6. Variation of *GAP* and *Time* for all algorithms when n_s changes.

n_s	IMS		ITS		ILV		ISV		BIS		SIDA^r	
	Gp	Time	Gp	Time	Gp	Time	Gp	Time	Gp	Time	Gp	Time
2	0.070	0.1075	0.090	0.0550	0.032	0.6291	0.122	0.0557	0.009	0.7916	0.211	0.0108
3	0.282	0.1076	0.283	0.0546	0.137	0.6301	0.332	0.0568	0.084	0.7923	0.328	0.0124
5	0.288	0.1089	0.286	0.0552	0.027	0.6378	0.232	0.0562	0.007	0.8018	0.403	0.0191
10	0.433	0.1524	0.430	0.0771	0.330	0.8834	0.416	0.0781	0.324	1.1130	0.136	0.0533
15	0.211	0.1544	0.209	0.0786	0.147	0.9000	0.213	0.0791	0.142	1.1330	0.126	0.0831
20	0.363	0.1586	0.362	0.0793	0.320	0.9289	0.363	0.0891	0.317	1.1669	0.095	0.1233
25	0.123	0.1610	0.132	0.0814	0.029	0.9420	0.138	0.0857	0.023	1.1845	0.178	0.1430
30	0.280	0.1639	0.279	0.0819	0.253	0.9542	0.286	0.0858	0.249	1.2000	0.052	0.1696
50	0.429	0.3003	0.432	0.1499	0.404	1.7788	0.430	0.1573	0.402	2.2290	0.050	0.5032

TABLE 7. Variation of *Gp* and *Time* for all algorithms when *Class* changes.

Class	IMS		ITS		ILV		ISV		BIS		SIDA^r	
	Gp	Time	Gp	Time	Gp	Time	Gp	Time	Gp	Time	Gp	Time
A	0.050	0.1432	0.046	0.0722	0.009	0.8385	0.045	0.0758	0.003	1.0540	0.119	0.0901
B	0.165	0.1458	0.165	0.0725	0.045	0.8523	0.161	0.0764	0.037	1.0706	0.214	0.0954
C	0.310	0.1470	0.319	0.0741	0.154	0.8589	0.294	0.0773	0.136	1.0801	0.257	0.1031
D	0.419	0.1440	0.431	0.0736	0.332	0.8456	0.464	0.0751	0.296	1.0632	0.217	0.0911
E	0.363	0.1443	0.360	0.0733	0.273	0.8415	0.376	0.0761	0.261	1.0591	0.178	0.0895

Table 7 that for the algorithm *BIS*, the highest gap values are obtained for classes C, D, and E. Contrarily, for the algorithm *SIDA^r* the highest gap values are obtained for classes B, C, and D.

Figure 5 illustrates the *Time* behavior for *BIS* when the pair (n_{SI}, n_s) changes.

VIII. DISCUSSION

Satellite imagery is increasingly used in various applications such as environmental monitoring, urban planning, agriculture, and disaster response. With the growing volume of satellite imagery data, managing and storing the data efficiently is becoming a significant challenge. Developing

effective storage support and used space management systems is crucial to address this challenge. In this paper, we proposed iterative-probabilistic-based algorithms that offer a promising solution for managing the used space for enhanced big satellite image management storage. Storage balancing is an effective solution for optimizing big data storage capacity and distribution, which involves monitoring, analyzing, and adjusting how data is stored to maximize cost savings and minimize downtime. Avoiding data bottlenecks by automatically redistributing data across multiple resources is crucial. One specific issue with big data storage is the effective transmission of large satellite images, such as satellite images, across a network to various storage supports. Finding an efficient way to schedule the satellite images to the storage supports that leads to equitable results in distribution is a challenge. This paper addresses this issue by proposing several heuristics and enhancement methods to solve the studied problem. The experimental study conducted in this paper shows that the proposed heuristics outperform those developed in the literature, with the *BIS* algorithm being the best in 73.87% of cases. While the proposed algorithms show promising results, there are potential limitations that should be considered. An important direction for future research could explore integrating machine learning and optimization techniques to enhance the proposed algorithms further. Integrating machine learning and optimization techniques can address some of the limitations of the proposed algorithms. For instance, machine learning algorithms can be trained to identify patterns in the data that can inform the optimization process and help to identify the best storage allocation and distribution strategies. Additionally, optimization techniques such as genetic algorithms or simulated annealing can be used to identify the best solutions through iterative testing. Combining machine learning and optimization techniques can improve the proposed algorithms' scalability and performance. For example, predictive models can be trained to identify the likelihood of specific satellite images being accessed or modified, which can be used to optimize storage allocation and distribution. Furthermore, anomalies or outliers in the data can be detected using machine learning algorithms to inform the optimization process and identify areas for improvement. In addition, in this paper, the development of a new lower bound is not treated, and as a result, it is not clear how close the proposed algorithms are to the optimal solution. Indeed, lower bounds are an essential tool in algorithm design and analysis; they provide a benchmark for measuring an algorithm's performance by indicating how far away the algorithm's optimal solution is. Future work in this area could focus on the development of a new lower bound that would allow for a more accurate comparison of the proposed algorithms' performance with the optimal solution. This would involve exploring the mathematical properties of the problem to derive a new lower bound that is both tight and efficient to compute. By having a new lower bound, it would be possible to quantify the gap between the proposed algorithms and the optimal solution. Additionally, it would allow for a comparison of the proposed algorithms with other methods in the literature, providing a more comprehensive evaluation and validation of the proposed algorithms.

IX. CONCLUSION

The paper proposes five novel heuristics to address the problem of assigning satellite images to storage supports. These heuristics are primarily based on enhancement phases. The first phase involves modifying the proposed probability distribution based on experimental results related to the randomization method. In the second phase, the initial algorithms are iterated multiple times to determine the best solution. The third phase allows for flexibility in selecting the biggest satellite images by considering the top three instead of just the two biggest ones.

Based on the experimental study, the proposed algorithm outperforms those found in the existing literature. For instance, in the case of *BIS*, the percentage of instances (out of 5300) is 73.87%, compared to 63.1% for $SIDA^r$. Additionally, the average gap for *BIS* is 0.147, while that of $SIDA^r$ is 0.197. Notably, the average running time of all proposed algorithms is acceptable when compared to the obtained results.

Consequently, the proposed algorithms offer the ability to develop an optimal solution for the problem under study. The field of heuristic-based scheduling of satellite images is constantly evolving, and future advancements are likely to be driven by improvements in technology, data analysis techniques, and optimization algorithms.

Future research includes exploring the use of swarm intelligence and quantum computing to optimize the scheduling of satellite images further. This involves investigating how the swarm intelligence algorithms can be integrated to explore the solution space and find improved scheduling solutions efficiently. Another promising future work will focus on exploring privacy-preserving techniques, such as zero-knowledge proofs, secure multi-party computation, or differential privacy, within the integrated ledger database and blockchain [39], [40], [41]. These techniques safeguard the confidentiality of sensitive scheduling data while leveraging the tamper-evident and auditable ledger database and the secure, decentralized nature of the blockchain. The main goal of using privacy-preserving techniques is to ensure privacy, data protection, transparency, and integrity in the scheduling system.

REFERENCES

[1] W. Ye, J. Lao, Y. Liu, C.-C. Chang, Z. Zhang, H. Li, and H. Zhou, "Pine pest detection using remote sensing satellite images combined with a multi-scale attention-UNet model," *Ecol. Informat.*, vol. 72, Dec. 2022, Art. no. 101906.

[2] W. Boulila, I. R. Farah, K. S. Ettabaa, B. Solaiman, and H. B. Ghézala, "Improving spatiotemporal change detection: A high level fusion approach for discovering uncertain knowledge from satellite image databases," in *Proc. Int. Conf. Data Mining*, vol. 9, Oct. 2009, pp. 222–227.

[3] A. Ferchichi, W. Boulila, and I. R. Farah, "Propagating aleatory and epistemic uncertainty in land cover change prediction process," *Ecol. Informat.*, vol. 37, pp. 24–37, Jan. 2017.

[4] Y. Afaq and A. Manocha, "Fog-inspired water resource analysis in urban areas from satellite images," *Ecol. Informat.*, vol. 64, Sep. 2021, Art. no. 101385.

[5] S. A. H. B. S. Muzamil, N. Y. Zainun, N. N. Ajman, N. Sulaiman, S. H. Khahro, M. M. Rohani, S. M. B. Mohd, and H. Ahmad, "Proposed framework for the flood disaster management cycle in Malaysia," *Sustainability*, vol. 14, no. 7, p. 4088, Mar. 2022.

[6] Z. Ayadi, W. Boulila, I. R. Farah, A. Leborgne, and P. Gançarski, "Resolution methods for constraint satisfaction problem in remote sensing field: A survey of static and dynamic algorithms," *Ecol. Informat.*, vol. 69, Jul. 2022, Art. no. 101607.

[7] N. R. Tadapaneni, "A survey of various load balancing algorithms in cloud computing," *Int. J. Sci. Advance Res. Technol.*, vol. 6, no. 4, pp. 484–487, 2020.

[8] V. Arulkumar and N. Bhalaji, "Performance analysis of nature inspired load balancing algorithm in cloud environment," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 3, pp. 3735–3742, Mar. 2021.

[9] A. Rafiq, M. S. Ali Muthanna, A. Muthanna, R. Alkanhel, W. A. M. Abdullah, and A. A. A. El-Latif, "Intelligent edge computing enabled reliable emergency data transmission and energy efficient offloading in 6TiSCH-based IIoT networks," *Sustain. Energy Technol. Assessments*, vol. 53, Oct. 2022, Art. no. 102492.

[10] T. Saba, A. Rehman, K. Haseeb, T. Alam, and G. Jeon, "Cloud-edge load balancing distributed protocol for IoE services using swarm intelligence," *Cluster Comput.*, vol. 2023, pp. 1–11, Jan. 2023.

[11] A. Kumar, A. Kaur, P. Singh, M. Driss, and W. Boulila, "Efficient multiclass classification using feature selection in high-dimensional datasets," *Electronics*, vol. 12, no. 10, p. 2290, May 2023.

[12] R. Singh, P. K. Gupta, P. Gupta, R. Malekian, B. T. Maharaj, D. Andriukaitis, A. Valinevicius, D. C. Bogatinoska, and A. Karadimce, "Load balancing of distributed servers in distributed file systems," in *Proc. Int. Conf. ICT Innov.* Cham, Switzerland: Springer, 2015, pp. 29–37.

[13] M. Adhikari and T. Amgoth, "Heuristic-based load-balancing algorithm for IaaS cloud," *Future Gener. Comput. Syst.*, vol. 81, pp. 156–165, Apr. 2018.

[14] A. Ragmani, A. El Omri, N. Abghour, K. Moussaid, and M. Rida, "A performed load balancing algorithm for public cloud computing using ant colony optimization," *Recent Patents Comput. Sci.*, vol. 11, no. 3, pp. 179–195, Nov. 2018.

[15] H. Alquhayz, M. Jemmali, and M. M. Otoom, "Dispatching-rule variants algorithms for used spaces of storage supports," *Discrete Dyn. Nature Soc.*, vol. 2020, pp. 1–9, Jun. 2020.

[16] L. Kong, J. P. B. Mapetu, and Z. Chen, "Heuristic load balancing based zero imbalance mechanism in cloud computing," *J. Grid Comput.*, vol. 18, no. 1, pp. 123–148, Mar. 2020.

[17] Y. Gupta, "Novel distributed load balancing algorithms in cloud storage," *Expert Syst. Appl.*, vol. 186, Dec. 2021, Art. no. 115713.

[18] T. C. Hung, L. N. Hieu, P. T. Hy, and N. X. Phi, "MMSIA: Improved max-min scheduling algorithm for load balancing on cloud computing," in *Proc. 3rd Int. Conf. Mach. Learn. Soft Comput.*, Jan. 2019, pp. 60–64.

[19] M. Alqarni, A. Cherif, and E. Alkayyal, "ODM-BCSA: An offloading decision-making framework based on binary cuckoo search algorithm for mobile edge computing," *Comput. Netw.*, vol. 226, May 2023, Art. no. 109647.

[20] D. R. Karger and M. Ruhl, "Simple efficient load-balancing algorithms for peer-to-peer systems," *Theory Comput. Syst.*, vol. 39, no. 6, pp. 787–804, Nov. 2006.

[21] M. Jemmali and H. Alquhayz, "Equity data distribution algorithms on identical routers," in *Proc. Int. Conf. Innov. Comput. Commun.* Cham, Switzerland: Springer, 2020, pp. 297–305.

[22] M. M. Hasan, S. Kwon, and J.-H. Na, "Adaptive mobility load balancing algorithm for LTE small-cell networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2205–2217, Apr. 2018.

[23] J. E. Rodriguez, "Systems and methods for load balancing using predictive routing," U.S. Patent 10 652 318, May 12, 2020.

[24] M. Jemmali, L. K. B. Melhim, S. O. B. Alharbi, and A. S. Bajahzar, "Lower bounds for gas turbines aircraft engines," *Commun. Math. Appl.*, vol. 10, no. 3, pp. 637–642, Sep. 2019.

[25] M. Jemmali, L. K. B. Melhim, and M. Alharbi, "Randomized-variants lower bounds for gas turbines aircraft engines," in *Proc. World Congr. Global Optim.* Cham, Switzerland: Springer, 2019, pp. 949–956.

[26] M. Jemmali, "Approximate solutions for the projects revenues assignment problem," *Commun. Math. Appl.*, vol. 10, no. 3, pp. 653–658, Sep. 2019.

[27] M. Jemmali, "Budgets balancing algorithms for the projects assignment," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 11, pp. 1–12, 2019.

[28] M. Alharbi and M. Jemmali, "Algorithms for investment project distribution on regions," *Comput. Intell. Neurosci.*, vol. 2020, pp. 1–13, Aug. 2020.

[29] M. Jemmali, "An optimal solution for the budgets assignment problem," *RAIRO-Oper. Res.*, vol. 55, no. 2, pp. 873–897, Mar. 2021.

[30] H. Alquhayz and M. Jemmali, "Max-min processors scheduling," *Inf. Technol. Control*, vol. 50, no. 1, pp. 5–12, Mar. 2021.

[31] M. Jemmali, L. K. B. Melhim, M. T. Alharbi, A. Bajahzar, and M. N. Omri, "Smart-parking management algorithms in smart city," *Sci. Rep.*, vol. 12, no. 1, pp. 1–15, Apr. 2022.

[32] M. Jemmali, L. K. B. Melhim, A. Alourani, and M. M. Alam, "Equity distribution of quality evaluation reports to doctors in health care organizations," *PeerJ Comput. Sci.*, vol. 8, p. e819, Jan. 2022.

[33] M. Jemmali, "Projects distribution algorithms for regional development," *Adv. Distrib. Comput. Artif. Intell. J.*, vol. 10, no. 3, pp. 293–305, Oct. 2021.

[34] M. Jemmali, "Intelligent algorithms and complex system for a smart parking for vaccine delivery center of COVID-19," *Complex Intell. Syst.*, vol. 8, no. 1, pp. 597–609, Feb. 2022.

[35] L. K. B. Melhim, M. Jemmali, and M. Alharbi, "Intelligent real-time intervention system applied in smart city," in *Proc. 21st Saudi Comput. Soc. Nat. Comput. Conf. (NCC)*, Apr. 2018, pp. 1–5.

[36] M. Jemmali, M. Alharbi, and L. K. B. Melhim, "Intelligent decision-making algorithm for supplier evaluation based on multi-criteria preferences," in *Proc. 1st Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, Apr. 2018, pp. 1–5.

[37] M. W. Shum, D. Wei, S. H. Wong, X. Y. Yang, and X. Zhou, "Dynamic load balancing for data allocation to servers," U.S. Patent 10 282 236, May 7, 2019.

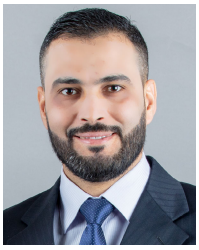
[38] N. Jain and G.-W. You, "Load balancing in cluster storage systems," U.S. Patent 8 886 781 B2, Nov. 11, 2014.

[39] D. Hasan and M. Driss, "SUBLμME: Secure blockchain as a service and microservices-based framework for IoT environments," in *Proc. IEEE/ACS 18th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Nov. 2021, pp. 1–9.

[40] A. Sadad, M. A. Khan, B. Ghaleb, F. Ali Khan, M. Driss, W. Boulila, and J. Ahmad, "Distributed twins in edge computing: Blockchain and IOTA," 2023, *arXiv:2305.07453*.

[41] W. Boulila, M. K. Khlifi, A. Ammar, A. Koubaa, B. Benjdira, and I. R. Farah, "A hybrid privacy-preserving deep learning approach for object classification in very high-resolution satellite images," *Remote Sens.*, vol. 14, no. 18, p. 4631, Sep. 2022.

MAHDI JEMMALI was born in Monastir, Tunisia. He received the engineering degree from the National Computer Science School of Tunisia, in 2003, the M.S. degree in mathematics engineering from the Polytechnic School of Tunisia, in 2005, and the Ph.D. degree in computer science from the University of Tunis, in 2011. He is currently an Associate Professor with the College of Computing and Informatics, University of Sharjah, Sharjah, United Arab Emirate. His research interests include algorithms, scheduling, big data, smart cities, operations management, and artificial intelligence. He is attracted by the application of several scheduling techniques to solve and implement real-life hard problems from different domains.



WADII BOULILA received the B.Eng. degree (Hons.) in computer science from the Aviation School of Borj El Amri, in 2005, the M.Sc. degree in computer science from the National School of Computer Science (ENSI), University of Manouba, Tunisia, in 2007, and the Ph.D. degree in computer science jointly from ENSI and Telecom-Bretagne, University of Rennes 1, France, in 2012. He is currently an Associate Professor of computer science with Prince Sultan University, Saudi Arabia. He is also a Senior Researcher with the RIOTU Laboratory, Prince Sultan University, a Senior Researcher with the RIADI Laboratory, University of Manouba, and previously a Senior Research Fellow with the ITI Department, University of Rennes 1. He has participated in numerous research and industrial-funded projects. His research interests include data science, computer vision, big data analytics, deep learning, cybersecurity, artificial intelligence, and uncertainty modeling. He is an ACM Member and a Senior Fellow of the Higher Education Academy (SFHEA), U.K. He received the Award of the Young Researcher in computer science in Tunisia from Beit El-Hikma, in 2021, the Award of Best Researcher from the University of Manouba, in 2021, and the Award of Most Cited Researcher from the University of Manouba, in 2021. He has served as the chair, a reviewer, and a TPC member for many leading international conferences and journals. His work has gained global recognition, and he has been nominated as one of the top 2% of scientists in his field by Stanford University.

ASMA CHERIF received the M.Sc. and Ph.D. degrees in computer science from Lorraine University, France, in 2008 and 2012, respectively. She conducted extensive research with the French Research Laboratory, Inria Nancy Grand Est. Since 2022, she has been leading the IoT Ecosystems Research Team, Center of Excellence in Smart Environment Research (CESER), King Abdulaziz University, Saudi Arabia. She is currently an Associate Professor with the Faculty of Computing and Information Technology, King Abdulaziz University. Her research interests include distributed systems and communication networks, collaborative applications, security, cloud/edge computing, computational intelligence, and the Internet of Things.

MAHA DRISS received the engineering degree (Hons.) in computer science and the M.Sc. degree from the National School of Computer Science (ENSI), University of Manouba, Tunisia, in 2006 and 2007, respectively, and the Ph.D. degree from the University of Rennes 1, France, in 2011. Throughout her career, she has held various academic roles. From 2012 to 2015, she was an Assistant Professor of computer science with the National Higher Engineering School of Tunis, University of Tunis, Tunisia. From 2015 to 2021, she was an Assistant Professor of computer science with the Information System Department, College of Computer Science and Engineering, Taibah University, Saudi Arabia. From 2021 to 2022, she held the position of Senior Researcher with the Security Engineering Laboratory, Prince Sultan University, Saudi Arabia. She is currently an Assistant Professor with the Computer Science Department, College of Computer and Information Sciences, Prince Sultan University. She is also a Senior Researcher with the RIADI Laboratory, University of Manouba. Her work has been published in reputable international journals and conferences. Her research interests include software engineering, service computing, distributed systems, the IoT, the IIoT, artificial intelligence, and security engineering. She is a member of the ACM Professional Chapter. She holds several professional certifications in cloud computing, service computing, and artificial intelligence. She has demonstrated her leadership capabilities by serving as the guest editor, the chair, a reviewer, and a TPC member for numerous leading international conferences and journals.