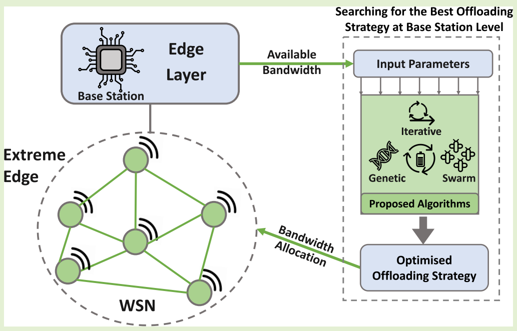
The Application of Evolutionary, Swarm, and Iterative-Based Task-Offloading Optimization for Battery Life Extension in Wireless Sensor Networks

Paula González, Gabriel Mujica, *Member, IEEE*, and Jorge Portilla, *Senior Member, IEEE*

***Abstract*—The proliferation of Internet-of-Things (IoT) devices has exponentially increased data generation, placing substantial computational demands on resource-constrained sensor nodes at the extreme edge. Task offloading presents a promising solution to tackle these challenges, enabling energy-aware and resource-efficient computing in wireless sensor networks (WSNs). Despite its recognized benefits, the exploration of task offloading in extreme edge environments remains limited in current research. This study aims to bridge the existing research gap by investigating the application of computational offloading in WSNs to reduce energy consumption. Our key contribution lies in the introduction of optimization algorithms explicitly designed for WSNs. Our proposal, focusing on bandwidth allocation, employs metaheuristic and iterative algorithms adapted to WSN characteristics, enhancing energy efficiency and network lifespan. Through extensive experimental analysis, our findings highlight the significant impact of task offloading on improving energy efficiency and overall system performance in extreme-edge IoT environments. Notably, we demonstrate a remarkable up to 135% reduction in network consumption when employing task offloading, compared to a network without offloading. Furthermore, our distinctive multiobjective approach, utilizing particle swarm algorithms, distinguishes itself from other proposed algorithms. This implementation effectively balances individual node consumption, resulting in an extended network lifetime while successfully achieving both specified objectives.**



***Index Terms*—Edge computing, extreme edge of Internet of Things (IoT), task offloading, wireless sensor networks (WSNs).**

I. Introduction

As TECHNOLOGY advances, electronic devices have become a ubiquitous part of people's daily lives, generating a constant stream of real-time data. This influx of information is a valuable resource in a wide range of applications, from medicine to strategic decision-making in business.

In this context, the Internet of Things (IoT) is poised to be one of the most transformative technologies of the 21st century. IoT presents itself as a logical and promising approach to addressing the challenges and opportunities of the digital age. It enables a wide range of devices and systems to be connected via the network, facilitating data collection, transmission, and analysis in real time. This further increases efficiency and adaptability in a wide range of applications [1], [2].

As shown in Fig. 1, the IoT has a layered structure that categorizes its component devices into different levels of complexity with their associated constraints. Although there is no unanimous agreement, it is widely accepted that the following four layers exist: cloud, fog, edge, and extreme edge [3]. Different computing strategies can be distinguished depending on the layer responsible for the data processing.

Cloud computing is based on offloading the processing of data collected at the edge to cloud servers. Its application to IoT brings some limitations in terms of latency, security issues, and reduced fault tolerance. In addition, the surge in data

Manuscript received 9 April 2024; accepted 17 June 2024. Date of publication 2 July 2024; date of current version 15 August 2024. This work was supported by the MCIN/AEI/10.13039/501100011033 through the Project TALENT-HIPSTER under Grant PID2020-116417RB-C41. The associate editor coordinating the review of this article and approving it for publication was Prof. Chen Yang. *(Corresponding author: Paula González.)*

The authors are with the Centro de Electrónica Industrial, Universidad Politécnica de Madrid, 28006 Madrid, Spain (e-mail: paula.gonzalezdedu@alumnos.upm.es; gabriel.mujica@upm.es; jorge.portilla@upm.es).

Digital Object Identifier 10.1109/JSEN.2024.3419558

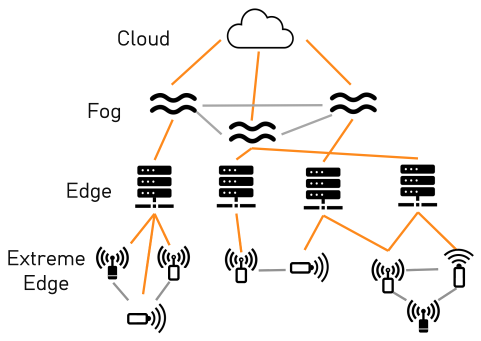


Fig. 1. Schematic overview of IoT layers.

collected can lead to network bottlenecks. Therefore, the trend in recent years has been to bring data processing closer to its source, giving rise to fog and edge computing [4]. These strategies succeed in reducing latency, therefore enabling real-time applications. However, they also introduce new constraints due to the limitations of the devices responsible for processing the data.

Unlike fog computing, edge computing is independent of the cloud, giving it autonomy in decision-making and enabling collaboration between IoT devices at the edge and the extreme edge participating in the wireless sensor networks (WSNs). These are networks of spatially distributed autonomous sensors placed at the extreme edge that communicate with each other and or a central control system at the edge or the extreme edge through wireless connections [5].

WSNs are widely used for monitoring and surveillance applications. However, the proliferation of IoT has led to the development of more complex applications that require advanced computing capabilities for sophisticated data analysis and decision-making. These applications implement advanced algorithms such as target recognition and object tracking, which require more processing power and energy consumption. This demand highlights the limitations of WSN sensor nodes, which are typically small and battery-powered. Therefore, a new challenge arises where strategies are needed to reduce the power consumption of these devices and thereby extend the lifetime of the networks [6].

Optimal sensor selection is crucial not only to improve system performance and reduce costs but also to overcome the energy constraints of the nodes. In their study on sensor selection for target recognition, Arora et al. [7] highlight the complexity of balancing rich sensor data with the limited computational resources of energy-constrained nodes in WSNs. They propose using less complex sensors, but this approach presents challenges in dealing with the imperfect nature of sensor outputs.

To the best of the authors' knowledge, there is limited research on the integration of task offloading in WSNs. Nonetheless, this strategic solution shows promise. By incorporating task offloading, nodes can potentially handle richer sensor outputs without compromising energy constraints. This allows the selection of sensors based on their suitability for their task rather than being limited by individual node capacities. Optimization of the overall performance of the WSN becomes achievable. Additionally, task offloading can be implemented through a collaborative approach, allowing devices to share resources for an equitable distribution that ensures optimization.

The proposal focuses on leveraging servers with high processing capacity in proximity to the network. These servers will engage in resource sharing by distributing idle computing resources among the devices, as discussed in prior surveys [8]. The efficacy of this resource-sharing strategy depends on various parameters, including the bandwidth of the base station (BS) and the volume of data exchanged over the network. Notably, our work delves into strategies aimed at optimizing bandwidth allocation in computational offloading scenarios, with a specific focus on edge IoT networks [9]. In these networks, characterized by devices closely located to data sources, resource-constrained sensor nodes actively participate in the WSN.

The primary focus of the proposed algorithms is to address the resource distribution challenges arising from applying task offloading in WSNs. The aim is to minimize energy consumption in WSNs while adapting to variations in the number of devices and the bandwidth shared between the network BS and the participating sensor nodes. To this end, we have tailored existing optimization algorithms, specifically genetic algorithms (GAs), particle swarm optimization (PSO) algorithms, and iterative methods, meticulously adapting them to suit the requirements of our resource distribution problem. The choice of these algorithms is justified by their proven effectiveness in handling complex optimization tasks, making them well-suited for addressing the complexities inherent in our computational offloading scenario involving variations in the number of devices and shared bandwidth between the network BS and sensor nodes.

In this way, the main contributions of this work are as follows.

1) Propose an underexplored approach for energy saving on WSNs by applying collaborative task-offloading strategies. This approach offers a promising route for extending the operational lifespan of sensor nodes.
2) Present a set of optimization strategies to select the most suitable offloading strategy throughout the network's lifecycle, with a primary goal of conserving battery power in the sensor nodes. These strategies are organized into two main approaches: iterative methods and metaheuristic algorithms (GA and PSO). Furthermore, each strategy includes a spectrum of variations for the proposed algorithms, enhancing adaptability. Extensive experimental analysis on WSN has been conducted, encompassing varied bandwidth scenarios and sensor node configurations.
3) Provide a selection guideline framework to aid in choosing the most suitable strategy based on the network's characteristics and the desired objectives.

The rest of the article is organized as follows. Section II presents state-of-the-art research on different strategies to apply computational offloading. Section III overviews the problem of optimizing the resource and bandwidth distribution in IoT sensor devices from the point of view of network lifetime, by applying the concept of computation offloading.

Section IV details the proposed offloading strategies to tackle the optimization problem set, while Section V outlines the main experimental results. The discussion of the results is done in Section VI. Finally, Section VII presents the conclusion and future lines of research for the proposed work.

II. Related Work

There are several trends in the state of the art addressing the optimization of resources at the edge of the IoT by offloading computational tasks of the nodes. In recent works, mobile edge computing (MEC) has been widely used as a strategy to implement task offloading in mobile networks. MEC provides an infrastructure at the edge that facilitates the efficient execution of computational tasks close to the end users by extending cloud computing services to the edge of networks, leveraging mobile BSs [10], [11].

Chen et al. [12] propose a multiuser task-offloading model for mobile-edge cloud computing. This paradigm provides cloud computing capabilities at the edge of radio access networks. One critical factor affecting offloading performance in this mechanism is the efficiency of wireless access. If many users simultaneously choose the same wireless channel to offload computation to the cloud, it may lead to low energy efficiency and data transmission time. In this case, offloading will not be beneficial. To achieve efficient offloading, they address two key challenges: 1) how should a mobile user choose between local and cloud computing? and 2) if a user chooses cloud computing, how to choose the right channel for high-efficiency wireless access? To address these challenges, they use a game theoretic approach and propose a distributed task-offloading algorithm to achieve the Nash equilibrium of the game.

Sardellitti et al. [13] design an algorithm to jointly optimize radio and computational resources for task offloading in a multicell MEC scenario, where a large amount of radio access points facilitates high bandwidth access to computational resources but increases intercell interference. The offloading problem objective is to minimize overall energy consumption at the mobile terminals under transmit power and latency constraints. The main challenge of this problem is considering the intercell interference, which makes the optimization problem non-convex. To solve this, they developed centralized and distributed SCA-based algorithms with provable convergence to locally optimal solutions of the non-convex problem. According to the results, the authors claim their proposed schemes converge faster and lead to significant energy savings compared to disjoint optimization procedures for applications requiring intensive computation and limited data exchange to enable offloading.

However, these proposals, while beneficial, are still dependent on the already saturated cloud and do not fully disengage from it. Additionally, MEC primarily targets mobile devices such as smartphones, which are more complex than the sensor nodes of WSNs. This means that, on the one hand, the computational tasks sent by MEC devices are more complex than those of the sensor nodes of WSN and thus require higher computing capacity at the BS (hence the need for cloud support in some circumstances). Additionally, the operating scheme is different from our objective. In MEC, the goal is to satisfy the needs of a client node, so once the task is computed for the client node, it must return to it. WSNs sought a collaborative computing scheme to achieve a single objective: to process the collected data and send it to the BS. This discrepancy in complexity and operational objectives leads to a closer look at opportunistic offloading as a complementary strategy that addresses the limitations of MEC.

Opportunistic offloading leverages edge computing resources, reducing the reliance on a centralized cloud infrastructure. The term opportunistic offloading represents a broad scheme that encompasses traffic and task offloading [14]. While traffic offloading focuses on reducing data redundancy and traffic load on the cellular networks, task offloading allows nodes with limited computing resources to offload tasks to nearby devices with idle computing capacity, extending battery life, increasing storage capacity, or improving application performance [15].

Shi et al. [16] suggest Serendipity, where mobile nodes can delegate their computational tasks to other mobile nodes through an opportunistic network. They model the computing tasks as PNP blocks. Each PNP block consists of three programs: pre-process, n-tasks that can be executed in parallel, and post-process. Pre and post-process programs must be executed locally, while the n-tasks are offloaded to neighbor nodes in parallel. Each node has a profile that describes its available capacities and is used to decide whether or not to allocate a task in an encounter node. If the client node does not receive the task result before a specified time, the offloading is discarded and the task is executed locally. They consider three models with different contact knowledge and control channel availability and design a task allocation algorithm for each.

Rehman et al. [17] propose an opportunistic task-offloading scheme to execute data mining tasks in mobile edge cloud computing (MECC) systems. They prioritize the execution of data mining tasks in mobile edge servers. However, when the resources are limited or there are no edge servers available, they offload the computation to the cloud. The algorithm works in three modes depending on which computational resources are used: local, edge servers, or the cloud. They propose a rule-based scheduling strategy to switch adaptively between different execution modes of the data mining algorithm. They develop their simulator for MECC-based data stream mining systems. Simulations show promising results for online mobile activity recognition applications. Yet, the proposed framework needs to be generalized.

However, these opportunistic offloading strategies, like MEC strategies, still focus on complex mobile networks. Furthermore, high node mobility is required to increase contact opportunities for these schemes to be effective. As mentioned above, this research focuses on WSNs in remote locations, where the probability of establishing contact with external devices is low. For this reason, opportunistic offloading strategies, while offering advantages in mobile networks, are not the most effective for WSNs.

To the best of our knowledge, a reduced number of works have addressed the study of the task-offloading problem in WSN. In such extreme edge conditions, energy conservation becomes critical due to the battery-power nature of the sensor nodes. These nodes, often located in challenging-access locations like forests, vast agricultural fields, or the depths of oceans and marine environments, impose constraints on communication bandwidth, battery life, and processing capacities.

Samie et al. [18] propose optimizing communication bandwidth by solving the so-called fragmentation issue. This problem highlights the underutilization of the gateway resources due to discrete coarse-grained offloading levels. They propose a switching method between different offloading levels at IoT devices such that it appears to the gateway as if the IoT device would operate at an intermediate offloading level and manage to reduce the battery consumption of the devices. However, this proposal does not consider the processing capability of the gateway and how its variation would affect the nodes.

A software-defined mission-critical wireless sensor network (MC-SDWSN) that can solve the existing challenging issues in traditional WSNs, such as resource utilization, data processing, system compatibility, and strict latency requirements, is proposed in [19]. Based on the MC-SDWSN architecture, they propose a centralized task-offloading algorithm to minimize the computing latency. They consider a network architecture similar to ours but assume that there is an infinite buffer in the edge servers. Their strategy consists of scheduling tasks by prioritizing those with the most critical latency.

Rong and Pedram [20] propose to reduce energy consumption on battery-powered mobile devices by task migration and remote processing based on Markovian decision processes. The dynamic power management problem is formulated as an optimization problem and solved using a linear programming approach. The experimental results prove the effectiveness of their methods. Their framework comprises clients, a server, and a wireless channel for communication. The server, assumed to be mains-powered with superior computational capability, handles local and remote tasks without energy and processing limitations. Clients initiate the offloading with remote process requests (RPRs) to the server with a time constraint. If the server cannot meet the specified time constraint, it rejects the request and the client must process the task locally, which wastes client resources. Hence, the offloading decision logic is integrated into the clients. This is viable for mobile nodes, but since this work focuses on resource-limited sensor nodes, migrating this logic to the server would eliminate the possible waste of the client's resources.

In conclusion, this study highlights the lack of research addressing the application of task offloading in WSNs as a strategy to reduce power consumption. As we look at the specific challenges of WSNs, particularly in remote environments, our focus narrows to the particular constraints and opportunities in this context. Notably, our work stands out in the WSN research for its unique emphasis on task offloading as a powerful strategy for reducing power consumption. In a landscape where energy efficiency is paramount, our contribution aims to fill a gap by providing optimization algorithms tailored to the specific needs of WSNs, thus contributing to their enhanced durability and efficiency.

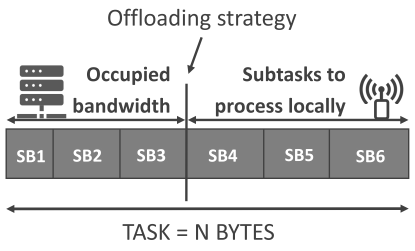


Fig. 2. Graphical depiction showcasing the division of tasks.

III. Problem Description and Objectives

This work compares different task-offloading algorithms in WSN infrastructures. The primary objective is to minimize battery consumption in extreme edge sensor nodes by applying task offloading. The strategy is to have edge servers that are connected to the power grid and have idle computational resources to make this capability available to the sensor nodes. By offloading some of the computational load from these nodes, it is possible to reduce their energy consumption and thereby maximize their lifetime.

An additional goal is set to make the battery consumption of the nodes as homogeneous as possible, favoring greater offloading in the nodes with lower batteries. In this way, if we consider that the network lasts until the first node runs out of battery, we will maximize the lifetime of the network. The way to achieve this goal varies depending on whether the algorithm is iterative or metaheuristic and is explained in more detail in Section IV.

Regarding the method for implementing the task-offloading strategy, a tiered system approach was selected. To define these levels, the task to be carried out by the nodes is established to be the same for all of them. This task has a fixed size of N bytes and can be divided into two fragments of variable size, each of which can be processed independently. Under this assumption, each level corresponds to a specific division of the task into two fragments, so that one of the parts is processed locally and the other is transmitted unprocessed to the BS to be handled by the BS. The levels defined used in the experimental analysis are detailed in Table I and Fig. 2 shows a graphical depiction of the division of a task that can be divided into six different size subtasks (SB).

This system can also be interpreted as each of the tasks can be divided into different independent processing subtasks. For example, if the task of the nodes is to capture the number of cars on a motorway at a certain time, the processing of the captured image could be divided into several subtasks such as image compression, pre-processing, and vehicle detection. Similarly, in healthcare applications involving the processing of ECG signals, tasks can be subdivided into distinct processing steps such as signal filtering, feature extraction, and anomaly detection. In this way, each level can relate to the decision of which subtasks are processed locally at the nodes and which are sent to the BS for processing. Due to the heterogeneity in the complexity of the subtasks, the definition of the processing cycles needed on each level has been done avoiding a linear progression.

Before describing the problem, it is necessary to define the network architecture under consideration.

TABLE I
LEVELS DESCRIPTION

Level	Processed Bytes	Cycles	Bandwidth
1	300	1,500,000	0
2	210	700,000	90
3	100	500,000	200
4	40	400,000	260
5	12	50,000	300

A. Network Architecture

We consider a local network of N geographically distributed IoT edge nodes $I = \{I_1, \ldots, I_N\}$ and a BS, which are connected through a local network with a star topology. A similar system model is considered in [18]. The BS is a peripheral server at the edge. It is considered that it performs several tasks, including helping the nodes process the data they collect. We use the term *bandwidth* in its data processing connotation to refer to the number of processing bytes that the edge server can share with the nodes in the network at any given time. As the BS is not exclusively dedicated to the network, we expect bandwidth to be a variable parameter over the lifetime of the network. The execution of the algorithms to obtain the offloading strategy will also be the responsibility of the BS. A similar strategy is also used by Li and Zhu [21] and Fu et al. [22], where they employ genetic algorithms (GAs), but the chromosomes in these algorithms take into account scheduling to reduce task completion time and power consumption, respectively. However, it is worth noting that those approaches demand higher computation from the edge device, which is a limitation in resource-constrained networks.

Two categories of parameters are defined to create node profiles: dynamic and static. Static parameters are linked to establish offloading levels and node hardware specifications which remain constant during network operation. In contrast, dynamic parameters reflect attributes of the nodes that evolve as they perform the tasks assigned by the BS. These parameters are

$$I_d = (Q_i, F, T_R, \mathrm{DC}, T_C, P_C, \mathrm{Bat}, b_{d_i}, E_d), d = 1, \ldots, N.$$

1) Static parameters.
   a) Q_i denotes the number of different offloading levels that the IoT sensor node can handle.
   b) F denotes the clock frequency of the node.
   c) T_R is the data transfer rate between the node and the BS, which must be set according to the characteristics of the network.
   d) *DC* represents the duty cycle of the node.
   e) T_C denotes the energy consumption during the data transmission from the node to the BS.
   f) P_C denotes the processing battery consumption of the nodes.
   g) *Bat* represents the initial battery of the nodes.
2) Dynamic parameters.
   a) b_{d_i} represents the number of bandwidth bytes occupied by node d working at level i.
   b) E_d is the remaining battery of sensor node d.

B. Optimization Problem

The problem considered in this work belongs to the class of generalized assignment problems (GAP). GAP consists of assigning a set of tasks to a set of agents with limited resources, taking into account a minimum total cost [23]. In this work, the resource is the bandwidth of the BS, which has to be allocated to the IoT nodes to minimize the total energy cost. All this applied to a network of IoT nodes sharing computational resources with a BS is called task offloading [24].

The task-offloading optimization problem can be formulated in many ways, as surveyed in [25]. For the description of the optimization problem of this work, the following definitions are needed.

1) B_d represents the overall occupied bandwidth of the network. Therefore, as shown in (1), it is obtained as the sum of the b_{d_i} of the nodes

$$B_d = \sum_{d=1}^{N} b_{d_i}. \tag{1}$$

2) P_d is the summation of the power consumption of all the nodes and is computed as shown in (2), where p_{d_i} is the total power consumption of device d operating at offloading level i

$$P_d = \sum_{d=1}^{N} p_{d_i}. \tag{2}$$

3) The computation of p_{d_i} of each node encompasses the on-board processing consumption p_{ob_i} and the consumption due to the data exchange transaction p_{tr_i} al level i. For the specific simulations of this work, p_{tr_i} is obtained based on single-hop communication, according to the considered star topology, as follows:

$$p_{\mathrm{ob}_i} = P_C \cdot \frac{\mathrm{Cycles}_i}{F} \tag{3}$$

$$p_{\mathrm{tr}_i} = T_C \cdot \frac{b_{d_i}}{\mathrm{TR}} \tag{4}$$

$$p_{d_i} = p_{\mathrm{ob}_i} + p_{\mathrm{tr}_i}. \tag{5}$$

Given these parameters, the optimization problem is defined by (6) and (7), where B_M stands for the maximum available bandwidth

$$\text{Optimization goal} : \min\,(P_d) \tag{6}$$

$$\text{Constraint} : B_d \leq B_M. \tag{7}$$

Equation 8 formulates the objective of maximizing the network's lifetime. Here, L_T represents the lifetime of the network, defined as the lowest battery value among the nodes. This objective is intentionally left implicit in some of the proposed algorithms to evaluate whether they inherently prioritize network lifetime optimization without explicit guidance. Thus, (8) is established as an additional objective

$$\text{Aditional Objetive} : \max\,(L_T). \tag{8}$$

1) During the simulations, T_R has been set in accordance with the IEEE Standard for Low-Rate Wireless

Networks (IEEE 802.15.4) in the 2.4-GHz band. This standard is widely used in WSNs. Nevertheless, this parameter can be configured in the proposed algorithms according to the used communication standard of the target WSN.

IV. Proposed Algorithms

Once the problem is defined, algorithms are developed to find the optimal bandwidth allocation and offload management routines throughout the network's dynamic lifetime. The proposed algorithms are designed to be topology agnostic and scalable, allowing them to adapt to variations in the number of nodes and available bandwidth. In addition, they share an initialization routine to help establish a common benchmark.

It is worth noting that although the algorithms can be applied to different network topologies, the obtention of the battery consumption of the node sending data assumes a single-hop communication link with the BS. Nevertheless, it is possible to transition to another type of network structure by adding the computation of the power consumption of the communication path accordingly, which allows the algorithms to operate at a higher optimization level.

In terms of network deployment, while optimal sensor placement (OSP) methodologies, such as those explored by Shi et al. [26], [27] and Yang and Ouyang [28], are crucial for addressing uncertainties in WSN and function as a deployment optimization phase, the proposed algorithms operate from a broader perspective. They aim to dynamically optimize the usage and distribution of resources in WSN, serving as a distinct phase from node placement. Thus, they work during network operation, seamlessly incorporating possible dynamic changes in WSN behavior.

We propose to solve the problem from two different perspectives: iterative and metaheuristic. Metaheuristic algorithms apply computational intelligence methods with advanced problem-solving capabilities to solve complex optimization problems [29]. In this work, we will focus on two types of metaheuristic algorithms that belong to the family of nature-inspired algorithms: GAs and PSO algorithms. GAs mimic evolution by searching large solution spaces, while PSOs, inspired by collective behavior, encourage collaborative exploration.

Conversely, iterative methods, while less complex, provide robust problem-solving capabilities through a sequence of incremental refinements. The decision to incorporate both metaheuristic and iterative techniques is strategic, as it allows for comparative analysis to identify the strengths and efficacy of each approach. Sections IV-A–IV-C comprehensively explain the proposed algorithms, highlighting their respective applications and strengths in the context of the optimization problem at hand.

A. Iterative Algorithms

The proposed iterative bandwidth allocation (IBA) algorithms use iterative strategies to distribute the available bandwidth among the nodes so that the maximum possible bandwidth is used, thereby minimizing the battery consumption of the IoT node. Two different methods are considered to carry out the bandwidth distribution, the iterative decremental allocation (IDA) and iterative incremental allocation (IIA) algorithms. Both IDA and IIA algorithms share the same input, information about the bandwidth and battery consumption of each level ordered from highest to lowest consumption in addition to the WSN characteristics.

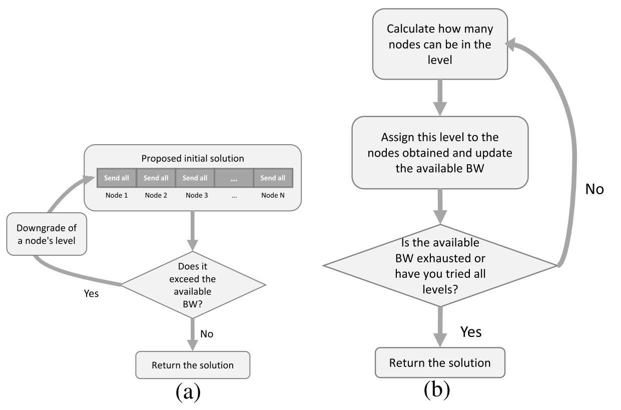

Proposed initial solution
Send all
Send all
Send all
...
Send all
Node 1
Node 2
Node 3
...
Node N
Downgrade of a node's level
Yes
Does it exceed the available BW?
No
Return the solution
(a)
Calculate how many nodes can be in the level
Assign this level to the nodes obtained and update the available BW
No
Is the available BW exhausted or have you tried all levels?
Yes
Return the solution
(b)

Fig. 3. Solution search procedure of the IBA algorithms. (a) IDA. (b) IIA.

1) As represented in the IDAs flowchart [Fig. 3(a)], this algorithm gradually adjusts an initial solution until the imposed bandwidth limit is satisfied. In the first step, it identifies the level with the lowest consumption and assigns all nodes to that level, thus forming the initial solution. It then checks whether this allocation exceeds the bandwidth limit. If it does, the algorithm iteratively reduces the level of a single node at each iteration, trying to meet the constraint. The resulting solution ensures adequate bandwidth distribution without exceeding the imposed constraint. This process is also described in Algorithm 1.
2) The IIA algorithm starts with an initial allocation where no node occupies bandwidth. The adjustment process is based on dividing the available processing bytes shared by the BS among the bandwidths of each level. Starting with the level that occupies the most space, the algorithm determines the maximum number of nodes that can operate on it. It then calculates the remaining available bandwidth and repeats the process for the next level with the second-highest byte usage. This sequence continues in descending order of levels until the available bandwidth is exhausted. The algorithm employs a greedy strategy to incrementally allocate bandwidth, optimizing resource usage at each step. The flowchart in Fig. 3 shows the process followed by this algorithm.

These iterative algorithms lack the ability to generate offloading strategies based on the state of the batteries in the network. Consequently, the execution of this type of algorithm divides the problem solution into two stages. First, the algorithm selects the optimal offloading strategy and then implements it, cyclically switching the levels of each node to achieve homogeneous network consumption. This process is repeated each time there is a variation in the available bandwidth, ensuring optimal redistribution.

Algorithm 1 Bandwidth Allocation IDA

```
Ensure: $sol$: List of the bandwidth allocation with the lowest consumption
 1: $sol \leftarrow []$
 2: $min \leftarrow$ None
 3: $min\_idx \leftarrow$ None
 4: $lev\_cons \leftarrow$ get_consumption(levels)
 5: for $idx, cons$ in ENUMERATE($lev\_cons$) do
 6:     if ($min$ is None or $cons < min$) then
 7:         $min\_idx \leftarrow idx$
 8:     end if
 9: end for
10: $sol \leftarrow n * [levels[min\_idx]]$
11: if SUM($sol$) $> BW$ then
12:     $nd \leftarrow 0$
13:     while SUM($sol$) $> BW$ do
14:         if $nd == n$ then
15:             $nd \leftarrow 0$
16:         end if
17:         $sol \leftarrow$ DOWNGRADE_LEVEL($sol$, $nd$)
18:         $nd \leftarrow nd + 1$
19:     end while
20: end if
21: return $sol$
```

B. Genetic Algorithms

GAs, first introduced by Jh [30], are a type of evolutionary strategy that belongs to the family of nature-inspired algorithms, which allow finding optimal solutions to complex problems. There is a wide variety of applications for which GAs are useful in MEC, one of them being bandwidth allocation. This can be seen in the study by Al-habob et al. [31], where they use a GA for scheduling sequential tasks in a multiserver network.

As depicted in Fig. 4, the foundation of a GA lies in the definition of chromosomes, as they represent the potential solutions. Historically, chromosomes were binary-coded. However, this approach posed limitations, particularly when addressing continuous optimization problems. To overcome these limitations, real-coded GAs (RCGAs) emerged. RCGAs represent solutions as vectors of real numbers, enabling a more natural and efficient exploration of continuous search spaces. The transition to RCGAs was motivated by the recognition that, for many real-world optimization problems, the flexibility and simplicity of using real numbers in chromosome representation outweighed the constraints imposed by binary coding [32].

The proposed algorithms are categorized as RCGAs, with their chromosomes representing the list of bandwidths to be occupied by each node in the network. In other words, each chromosome embodies a specific offloading strategy, with its size corresponding to the number of nodes in the network. Fig. 5 illustrates the structure of these chromosomes.

The initial population consists of randomly generated chromosomes that evolve through crossover and mutation over several generations until an optimal solution is reached. Random generation of the initial population is essential to ensure maximum diversity. However, despite this randomness, all chromosomes must obey the constraint outlined in (7). In addition, to maintain said diversity, the population size must be sufficiently large, a parameter whose selection is considered in Section V.

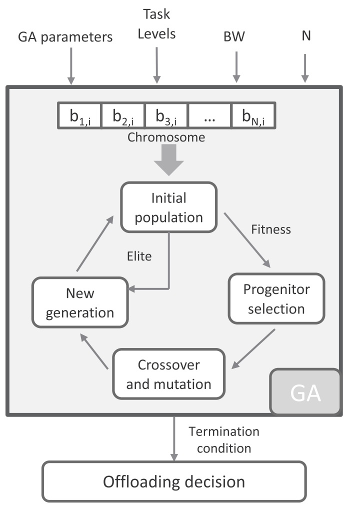


Fig. 4. Outline of the process followed by the GAs.

b_{1_i}	b_{2_i}	...	b_{N_i}

Fig. 5. Chromosome description.

The next step is to select the most suitable individuals for mating, as determined by their fitness level. This fitness level is assessed using a fitness function, which calculates a score indicating an individual's likelihood of reproducing. The effectiveness of several proposed fitness functions is evaluated in the following section.

As shown in [33], there are several ways to implement the selection of the individuals. After considering the various options, we decided that selection by tournament was best suited to the problem at hand. The first step in Tournament Selection is to randomly select a set of individuals. These individuals are ranked according to their calculated fitness, and then the fittest one is selected for mating. The number of individuals selected for each tournament is set to two. Since the descendants come from the crossing of two parents, this process is repeated twice for each time a new chromosome is to be generated.

Once the progenitors have been selected, the mating process will begin. The mating is done by single-point crossover. For the problem formulated, none of the solutions must exceed the bandwidth. Thus, after the generation of the children, they are checked for compliance with this constraint and the offspring that do not satisfy it are discarded. The process of selecting and mating the parents continues until a new generation of the given population size is obtained.

The last step is mutation. The mutation maintains diversity in the population, preventing the solution from stagnating at a local optimum. As in Darwin's theory, mutation applies to a random number of individuals. In our algorithms, the probability of an individual being selected for mutation is an adjustable parameter. Once an individual has been selected for mutation, the mutation routine selects a random node and changes its bandwidth to that of another random level. The mutated individual may exceed the bandwidth. Given this case, this individual is discarded and the original individual is reintroduced into the population. This implies a reduction in the previously set mutation rate. It has been experimentally determined that the number of mutations rejected due to noncompliance with this constraint does not exceed 30%. This is taken into account in the selection of the GA experimental parameters, where it has been decided to set a higher value for the mutation rate to compensate for this effect.

The proposed algorithms incorporate elitism. Elitism is an operational feature of GAs that provides a means of reducing genetic drift by ensuring that the best chromosome is passed without modification to the next generation [34]. In this way, the best individual of the current generation called the *elite* is passed on to the next generation. There are several ways in which the elite can be incorporated. This article considers two of them. GA-mean consumption (GA-M) and GA-lifetime (GA-LT) add the elite to the next generation when the mutation of its offspring is complete. GA-hybrid (GA-H) replaces the worst individual in the next generation with the elite. Furthermore, each of these algorithms uses a different elite selection criteria.

The new generation completes once elitism has been performed. The number of generations must be large enough to allow the GAs to reach the best solution and can be determined empirically. Another option is to add a termination condition that stops the creation of new generations when a high percentage of the population satisfies it. To ensure that the number of generations is sufficient, all three proposed algorithms have a termination condition that stops the creation of new generations when more than 80% of the population has the same fitness.

The solution given by the GAs is the chromosome with the best fitness in the last generation. In most cases, there will be more than one individual with the best fitness. Selecting the final solution from these individuals is similar to selecting the elite chromosome and is discussed below.

Three different GAs have been developed: GA-M, GA-LT, and GA-H. These algorithms differ in their fitness functions and the method used to select the elite individual as summarized in Table II.

GA-M and GA-H prioritize minimizing the average network consumption as their main objective, defined in (6). Consequently, the best chromosomes have the lowest Fitness$_M$. In GA-LT, a more homogeneous network consumption is prioritized. The fitness of the chromosomes is set as the minimum battery capacity the network would require if this solution were chosen, and following the objective in (8), the algorithm seeks solutions with higher Fitness$_{\mathrm{LT}}$.

GA-M and GA-LT employ the same elitism method, where the first individual in the list with the highest fitness is selected as the elite and is added directly to the next generation. GA-H is an update to GA-M that purports to achieve a lower standard deviation between node batteries by modifying the elitism method. Instead of selecting the elite as the first chromosome in the list with the highest fitness, the elite is selected as the one with the highest minimum network battery among all individuals with the highest fitness. Additionally, GA-H replaces the worst individual in the next generation with the elite of the current one, to get to the best solution faster.

TABLE II
GA SPECIFICATIONS

Algorithm	Fitness Function	Elitism
GA-M	$Fitness_M = \frac{P_d}{N}$	Random chromosome within the ones with lowest $Fitness_M$
GA-LT	$Fitness_{LT} = L_T$	Random chromosome within the the ones with highest $Fitness_{LT}$
GA-H	$Fitness_M = \frac{P_d}{N}$	Chromosome with highest L_T within the he ones with lowest $Fitness_M$

The functional behavior of GAs is different from the one of iterative algorithms. Instead of only finding a new solution when the bandwidth is modified, GAs are designed to discover the best solution for the current network configuration at each of the execution's DC. This distinction results from the compatibility of GAs for dynamic networks, where nodes possess varying initial battery levels, and the number of nodes can fluctuate. GA-M and GA-LT are executed through the two functions represented in Algorithm 2, and GA-H in Algorithm 3.

Algorithm 2 GA for GA-M and GA-LT

```
procedure NEXTGENERATION($currentGen$, $mutation\_rate$)
    $fitness\_list$ ←GET_ALL_FITNESS($currentGen$)
    $best\_idx$ ←GET_BESTFIT_IDX($fitness\_list$)
    $elite$ ←$currentGen[best\_idx]$
    $children$ ←MATING($currentGen$, $fitness\_list$)
    $nextGen$ ←MUTATE_POPULATION($children$, $mutation\_rate$, $elite$)
    return $nextGen$
end procedure

procedure GENETICALGORITHM($mutation\_rate$, $popSize$, $generations$)
    $pop$ ← GENESIS($popSize$)
    for $i$ ← 1 to $generations$ do
        $pop$ ← NEXTGENERATION($pop$, $mutation\_rate$)
        $fitness\_list$ ←GET_ALL_FITNESS($pop$)
        $max$ ← $None$
        for $num$ in $fitness\_list$ do
            if $max$ is None or $max < num$ then
                $max \leftarrow num$
            end if
        end for
        $index$ ← FIND_INDEX($fitness\_list$, $max$)
        if $len(index) \geq popSize * cond\_fin$ then
            break
        end if
    end for
    $best\_sol$ ← []
    for $i$ ← 1 to $len(index)$ do
        $best\_sol$.append($pop[index[i]]$)
    end for
    return $best\_sol[0]$, $len(index)$
end procedure
```

Algorithm 3 GA for GA-H

```
procedure NEXTGENERATION($currentGen$, $fitness\_list$, $mutation\_rate$, $elite$, $elite\_fit$)
    $children$ ←MATING($currentGen$, $fitness\_list$)
    $nextGen$ ←MUTATE_POPULATION($children$,$mutation\_rate$)
    $next\_fitness\_list \leftarrow get\_all\_fitness(nextGen)$
    $worst\_idx \leftarrow get\_worstfit\_idx(next\_fitness\_list)$
    $nextGen[worst\_idx] \leftarrow elite$
    $next\_fitness\_list[worst\_idx] \leftarrow elite\_fit$
    return $nextGen$, $next\_fitness\_list$
end procedure

procedure GENETICALGORITHM($mutation\_rate$, $popSize$, $generations$)
    $pop$ ← GENESIS($popSize$)
    $fitness\_list$ ← GET_ALL_FITNESS(pop)
    for $i \leftarrow 1$ to $generations$ do
        $elite, elite\_fit, repeated$ ←FIND_ELITE($pop$, $fitness\_list$)
        $pop$, $fitness\_list$ ←NEXTGENERATION($pop$, $fitness\_list$, $mutation\_rate$, $elite$, $elite\_fit$)
        if $len(index) \geq popSize * cond\_fin$ then
            break
        end if
    end for
    return $nextGen$, $next\_fitness\_list$
end procedure
```

C. Particle Swarm Optimization Algorithms

PSO algorithms, first proposed by Kennedy and Eberhart [35] and inspired by the flocking behavior of birds, use exploration techniques to search for parameters that optimize a defined objective. Their origin is based on two concepts, swarm intelligence and evolutionary computation [36].

The operation of PSO algorithms is based on the movement of a swarm of particles through a *D-dimensional* search space to find an optimal solution. The position of each particle defines a proposed solution. The particles are defined by a current position vector $X_i = (x_{i1}, x_{i1}, \ldots, x_{iD})$ and its current velocity vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, where D is the number of dimensions [37].

The algorithm starts by randomly initializing V_i and X_i. Then, at each iteration, the best position found by each particle i, $P_{\text{best}_i} = (P_{\text{best}_{i1}}, P_{\text{best}_{i2}}, \ldots, P_{\text{best}_{iD}})$ and the best position found by the whole swarm $G_{\text{best}} = (G_{\text{best}_1}, G_{\text{best}_2}, \ldots, G_{\text{best}_D},)$ both guide the particle i by updating its velocity and position using the following equations:

$$v_{i_d}(t+1) = w \cdot v_{i_d}(t) + C1 \cdot r1 \cdot \left(P_{\text{best}_{id}}(t) - x_{i_d}(t)\right) + C2 \cdot r2 \cdot \left(G_{\text{best}_d}(t) - x_{i_d}(t)\right) \tag{9}$$

$$x_{i_d}(t+1) = x_{i_d}(t) + v_{i_d} \cdot (t+1). \tag{10}$$

1) ***w***: It represents the *inertia weight*, which controls the effect of the particle's current velocity on its new velocity. Selecting this parameter adjusts the local and global search capabilities. If $w = 0$, the new velocity of the particle is independent of the previous one, this means there is no memory involved and that it only depends on the best local and global (P_{best} y G_{best}). Conversely, when $w \neq 0$, the tendency of the particle to explore new positions is inspired. This tendency grows as the value of w increases, resulting in longer flight steps. On the other hand, decreasing w will reduce the flight speed, causing the particles to focus on local exploration.
The value of *inertia weight* can be fixed or given a linearly decreasing weight (LDW). This method encourages global exploration during the first iterations of the algorithm and shifts the focus to local exploration as the particles approach the optimal solution. Its implementation follows the formula:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{T_{\max}} \cdot t \tag{11}$$

where $w_{\max}$ is the maximum value of the *inertia weight* and $w_{\min}$ is the minimum; t represents the current iteration and $T_{\max}$ the total number of iterations.
2) ***C1*** and ***C2***: These are the acceleration coefficients and represent the weight of the stochastic acceleration of each particle toward its historical best position (P_{best}) and the best global position (G_{best}), respectively. C1 is the cognitive parameter, which represents the maximum influence of the particle's best position on its new velocity, and C2 is the social parameter and indicates the maximum influence of social information on the new value of the particle's velocity (G_{best}). Both usually have values close to two.
3) ***r1*** and ***r2***: These are random values between 0 and 1.

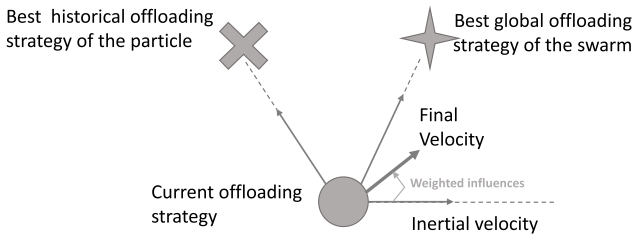

Best historical offloading strategy of the particle
Best global offloading strategy of the swarm
Final Velocity
Weighted influences
Current offloading strategy
Inertial velocity
Fig. 6. Particle's velocity representation.

Fig. 6 represents graphically how the best local and global positions affect the calculation of the new velocity of a particle.

Specific to the problem posed, it is established that each dimension of the search space corresponds to the bandwidth occupied by each node in the network, giving a total of D dimensions, equal to the number of nodes present. Due to the nature of the problem, the range of values that each dimension can take is discrete and must coincide with the bandwidth of each offloading level. In this way, each position in the search space represents a unique computational offloading strategy, and the velocities of the particles determine whether the node increases decreases, or maintains its level.

Being a discrete space, it is necessary to introduce certain adaptations to the PSO related to updating the positions of the particles. Velocity values obtained by (9) are not integers and using them to update the particle positions ((10)) can result in bandwidths that most probably would not correspond to any of the predefined levels. To deal with this situation, a procedure is established whereby, once the new particle position is obtained, the bandwidth of each node must be approximated to the nearest available level.

Another essential adjustment when applying PSO is related to the restraint (7), which states that the sum of the bandwidths occupied by all nodes in the network must not exceed the available bandwidth. In the first approach, it was found that if the new position of a particle exceeded the bandwidth limit, the calculation of its velocity had to be repeated until this condition was met. Although this alternative proved satisfactory for networks with five nodes, as the number of nodes increased, the algorithm became stuck, preventing the completion of the execution program. After an exhaustive search for solutions, an approach was found in the work of [38] that penalizes the fitness of those particles that do not satisfy the constraint. This method proved to be very compatible with the problem at hand. Consequently, it is decided that particles exceeding the bandwidth constraint will be assigned a fitness of zero. This approach does not affect either the P_{best} or the G_{best}, while the exploration of the space is unaffected.

Algorithm 4 presents the steps followed by the PSO algorithms.

Algorithm 4 Particle Swarm Algorithm

```
1: function $PSOalgorithm$
2:     $particles, vel \leftarrow$ genesis($num\_part\_2$)
3:     $part\_fitness \leftarrow$ get_all_fitness($particles$)
4:     $pBest \leftarrow [[0] * n] \times num\_part\_2$
5:     $pBest\_fit \leftarrow [[0] * num\_part\_2]$
6:     $gBest \leftarrow []$
7:     $gBest\_fit \leftarrow 0$
8:     for $i \leftarrow 1$ to $generations$ do
9:         $w \leftarrow Parameters.w - i \times (Parameters.w/generations2)$
10:        for $j$ in $[0, num\_part\_2)$ do
11:            if $part\_fitness[j] > pBest\_fit[j]$ then
12:                $pBest[j] \leftarrow particles[j]$
13:                $pBest\_fit[j] \leftarrow part\_fitness[j]$
14:            end if
15:            if $part\_fitness[j] > gBest\_fit$ then
16:                $gBest \leftarrow particles[j]$
17:                $gBest\_fit \leftarrow part\_fitness[j]$
18:            end if
19:        end for
20:        $new\_pop \leftarrow []$
21:        $new\_vel \leftarrow []$
22:        $aux \leftarrow 0$
23:        while $aux < num\_part\_2$ do
24:            $new\_part, new\_vel\_part \leftarrow$ get_new_part($particles[aux]$, $vel[aux]$, $pBest[aux]$, $gBest$, $w$)
25:            $new\_pop$.append($new\_part$)
26:            $new\_vel$.append($new\_vel\_part$)
27:            $aux \leftarrow aux + 1$
28:        end while
29:        $particles \leftarrow new\_pop$
30:        $part\_fitness \leftarrow$ get_all_fitness($particles$)
31:        $vel \leftarrow new\_vel$
32:    end for
33:    for $i$ in $[0, num\_part\_2)$ do
34:        if $part\_fitness[i] > gBest\_fit$ then
35:            $gBest \leftarrow particles[i]$
36:            $gBest\_fit \leftarrow part\_fitness[i]$
37:        end if
38:    end for
39:    return $gBest$
40: end function
```

Similar to the GAs, three particle swarm algorithms have been developed, PSO-mean consumption (PSO-M), PSO-lifetime (PSO-LT), and PSO-hybrid (PSO-H), whose differences lie in the way of calculating the fitness of the particles. All three share the previously defined topology of the search space and the representation of the particles. Table III summarizes their fitness functions.

TABLE III
PSO ALGORITHMS SPECIFICATIONS

Algorithm	Fitness Function
PSO-M	$Fitness_M = \frac{P_d}{N}$
PSO-LT	$Fitness_{LT} = L_T$
PSO-H	$Fitness_H = \lambda * Fitness_M + (1 - \lambda) * Fitness_{LT}$

Algorithms PSO-M and PSO-LT are programmed to investigate whether the space exploration approach of PSO algorithms offers advantages compared to the evolutionary process followed by GAs. To this end, it is established that PSO-M shares the same fitness function as GA-M and GA-H (Fitness_M) and that PSO-LT shares that of GA-LT ($\text{Fitness}_{\text{LT}}$). That is, in PSO-M, the fitness of a particle coincides with the average consumption of the network that would result from the solution proposed by that particle. In the PSO-LT algorithm, the fitness is the lowest residual battery among the nodes after executing the task using the strategy contained in the particle.

PSO-H introduces a different way of obtaining the fitness of particles based on multiobjective optimization. Multiobjective optimization problems (MOP) involve computing multiple functions with conflicting objectives. Cui et al. [39] provide a detailed overview of multiobjective optimization methods and their application in energy conservation, including trade-off methods. Trade-off methods involve converting these complex problems into single-objective problems, which can then be solved using classical optimization algorithms. To use this approach, it is necessary to determine the relative importance of each objective, either by using the weighted sum method (WSM) beforehand or by using an iterative method during the search process. After evaluating these methods, the best strategy for the defined problem is to use a compensation method based on WSM.

Up to this point, the algorithms have been oriented toward one of two stated objectives: minimizing the consumption of the network or maximizing its remaining lifetime. As the simulations will show, the minimization-oriented methodology fails to achieve a balance in consumption, while the maximization-oriented methodology achieves a fair distribution, but at the expense of the average consumption. PSO-H aims to combine both objectives. The fitness function of PSO-H is based on the work of Wu et al. [40] and consists of a weighted sum between the two fitnesses defined by (12), where λ is the weighting factor.

Through empirical experimentation, it has been found that a lambda value of 0.6 effectively establishes a trade-off between minimizing battery consumption and controlling standard deviation. Prioritizing minimization with a lambda value higher than 0.6 may lead to lower energy use but a higher standard deviation. Conversely, lower lambda values prioritize controlling standard deviation, potentially resulting in a more homogeneous distribution but increased battery consumption.

While 0.5 might seem like an ideal midpoint between these objectives, research suggests that a slightly higher value of 0.6 achieves a similar balance. Moreover, it reduces time complexity by enabling quicker convergence to the optimal solution. This optimized point ensures efficient energy usage while maintaining balance, offering a practical compromise between the two objectives

$$\text{Fitness}_H = \lambda * \text{Fitness}_M + (1 - \lambda) * \text{Fitness}_{\text{LT}}. \quad (12)$$

It is important to note that this fitness function was implemented in the GAs. However, the results obtained did not show any significant improvement concerning the algorithms defined in this work.

As far as the execution methodology of the PSO algorithms is concerned, the same procedure is used for the GA algorithms and is detailed in Section V.

V. Configuration and Parameter Selection for Experimental Analysis

Several experiments have been carried out under common network conditions to test and compare the efficiency of the implemented algorithms in terms of average battery consumption, standard deviation, and time complexity. Extensive simulations covering a wide range of network conditions were performed using a device with a 2.3-GHz Intel Core i5 dual-core processor and 8 GB of RAM.

The initial phase of establishing the experimental environment involved defining the characteristics of the network. Table IV details the selected parameters for both nodes and the BS. These parameters are configured based on a Cookie modular node which comprises a C8051-based processing layer and a CC2420-based communication layer. Subsequently, it is necessary to define both the test bench and the parameters to evaluate the algorithms' performance. In terms of energy efficiency, it is decided to measure the average energy consumption of the network. Standard deviation is selected as a measurement of the homogeneity of the nodes' batteries. Standard deviation quantifies the dispersion of values with the mean, thus providing a solid metric for assessing the fairness of the energy distribution between nodes.

In addition, measuring the execution time of the algorithms will allow for assessing the feasibility of applying the algorithms in real-live network environments, where time complexity is a critical factor. These measures would provide a comprehensive view of the performance of the algorithms, generating valuable information for deciding which alternative to use for a given application.

Concerning the offloading levels, the task has a size of 300 bytes and is divided into two parts in five different ways corresponding to the five levels in Table I. Each level contains information about the number of bytes sent to the BS for processing (bandwidth) and the number of processing cycles the sensor node needs to process the rest of the task. As discussed in Section III, for the experiments to represent conditions close to reality, neither the bandwidth nor the processing cycles change linearly from one level to the next. It is worth noting that at level 5, where the node sends the entire task to the BS, a small amount of local processing of the collected data is considered before sending it. This accounts for the consumption associated with preparing the data for transmission.

TABLE IV
Parameters of the Network

Parameter	Value
Frequency	8 MHz
Transmission rate (TR)	250 Kbps
Transmission consumption (T_C)	10 mA
Processing consumption (P_C)	4 mA
Battery	12000 mAh

The simulation has a time-based structure defined by a DC, which symbolizes the maximum time required for the BS to receive the order to execute a task, determine the appropriate task-offloading strategy, transmit the instructions to the nodes, and finally complete the task and send it to the BS. The simulation program iterates the DC K times, with K being a configurable parameter.

In developing the algorithms' execution method, as described in Section IV, there is a distinction between iterative and metaheuristic approaches. The distinction comes from the ability to adapt to the environment inherent to metaheuristic methods. In contrast, iterative algorithms face difficulties in considering the state of the nodes' batteries in the search for solutions, which could affect the network lifetime in situations where there is inequality in the nodes' batteries. As noted above, the operation of the iterative algorithms only updates the bandwidth allocation when the available bandwidth changes and then performs the cyclic switching of this solution to achieve an equal distribution of consumption, as discussed in Section IV.

The execution strategy followed by the metaheuristic algorithms requests a new bandwidth allocation at each DC, generated based on the number of nodes and their batteries, which are updated after each cycle. This way, since the GAs have variable-length chromosomes and the dimensions of the search space in the PSO algorithms are an easily varied parameter, the metaheuristic algorithms can adapt to any failure in the network's nodes.

It is important to note that although this dynamic may affect the ability of metaheuristics to achieve the minimum standard deviation, it also makes them more attractive in environments with dynamic networks, where the number of nodes in the network and their parameters may change during its lifetime. In addition, when metaheuristic algorithms are executed with the same method as the iterative algorithms, they obtain identical average battery consumptions and differ from the results shown in this work only in the standard deviation of the batteries, which is zero for all algorithms.

In the light of the above, the following assumptions are made.

1) The first analysis is done with a 20-node network, which is extended to 50–100 nodes in the following experiments.
2) Since the iterative algorithms do not take into account the state of the network to obtain the offloading strategies, it is decided to start from the most favorable situation for the iterative algorithms. Therefore, all nodes

have the same initial battery, that of Table IV. Moreover, in this way, it is possible to see whether the metaheuristic algorithms can outperform the iterative algorithms when they compete in the most favorable conditions of the latter.

3) As previously stated, due to the BS not being exclusively dedicated to the network, bandwidth is expected to vary throughout its lifespan. Consequently, a diverse set of scenarios is considered during the experimental tests, generating data for 21 different bandwidths. These bandwidths progressively increase from a minimum level, where only 5% of the nodes can operate at the maximum sending level (level 5), up to 95.
4) K is set to 100 DCs.
5) The parameters of the genetic and particle swarm algorithms are kept constant throughout the different network analyses. The purpose of keeping the parameters constant is to assess the scalability of the metaheuristic algorithms when the network scales. In this way, the aim is to determine the ability of the algorithms to maintain the reduction in energy consumption when, already implemented in a network, the number of nodes varies.
6) The experimental results are displayed in three different plots.
   a) Mean battery consumption.
   b) Standard deviation between the batteries of the network after the execution for each bandwidth, to measure the homogeneity of the battery consumption solution.
   c) Average execution time of the algorithms.

The calibration of the parameters of the GAs is crucial for optimal algorithm performance. Gibss et al. [41] propose a series of methods that attempt to provide insight into how to set these values. Of those described, the method selected for this work is trial and error. The assumptions made to establish the parameters of the GAs are as follows.

1) *Population size*: The size of the population is a critical factor in the performance of an algorithm. If the number of individuals is too small, the algorithm may prematurely converge to a suboptimal solution. On the other hand, a large population may lead to unnecessary computational and resource usage without guaranteeing improvement in results. After considering this trade-off, empirical analysis has determined that a population size of 200 chromosomes is optimal for the problem at hand.
2) *Number of generations*: The criticality of the number of generations decreases when a termination condition is included. The termination condition is set for all GAs so that the algorithms stop generating new generations when more than 80% of the chromosomes have the same fitness or the algorithm has performed 100 generations to avoid blockages in case the algorithm fails to find the optimal solution.
3) *Mutation rate*: The mutation rate on GAs is usually set to low values, such as 0.01. However, experimental findings indicate that setting the mutation rate to 0.1 significantly improves the average consumption and the standard deviation. The value of the mutation rate has been chosen considering the reduction of the mutated individuals due to the bandwidth restriction.

As with GAs, the correct choice of parameters is essential for particle swarm algorithms to achieve their full potential. The values of the parameters for the PSO algorithms are as follows.

1) *Particle number:* The correct choice of the number of particles is crucial for optimal performance. There is no universal range for the number of particles, as this value varies with the type and complexity of the algorithm. However, many works follow the suggestion on [35]. They recommend using a population size between 20 and 50 particles. Piotrowski et al. [42] explore the impact of swarm size on the performance of several variants of the PSO algorithms. This work concludes that the range of 20-50 particles is relatively small for new variants of PSO, which typically require larger populations of 70-500 particles. After carrying out the corresponding tests with the presented algorithms and considering their greater similarity to the classical PSO, it is concluded that a population of *20 particles* adequately satisfies the stated objectives.
2) *Number of generations:* As with GAs, the importance of the number of generations is reduced significantly by incorporating a termination condition. The same condition has been chosen for the PSO, stopping the execution when more than **80%** of the particles are in positions with the same fitness or when the algorithm has completed 100 generations.
3) *Inertia weight, w:* As discussed in Section IV-C, the inertia weight can either take a constant value or be varied as a function of an LDW as shown in 11. The second approach has been shown experimentally to be more efficient. Therefore, **w** is determined to gradually decrease in value from **1.2** to **0.6** over the iterations of the PSO.
4) *Cognitive parameter (C1) and social parameter (C2):* The acceleration constants, C1 and C2, are set to **1.9** and **0.3**, respectively. Empirical experimentation reveals that lowering exploration (C1) can trap the algorithm in local optima in the considered rugged search space while increasing exploitation (C2) does not enhance algorithm performance.

VI. Analysis and Interpretation of Experimental Outcomes

A. Mean Consumption Results

The average consumption results of the eight algorithms, simulated across various bandwidths on networks with 20, 50, and 100 nodes, are presented in Fig. 7. Notably, distinguishing between GA-M and GA-H in all plots is challenging due to their nearly identical average consumption, attributed to both algorithms sharing the same fitness function.

In contrast, comparing the maximum and minimum average consumption values shown in Fig. 7(b), with the consumption

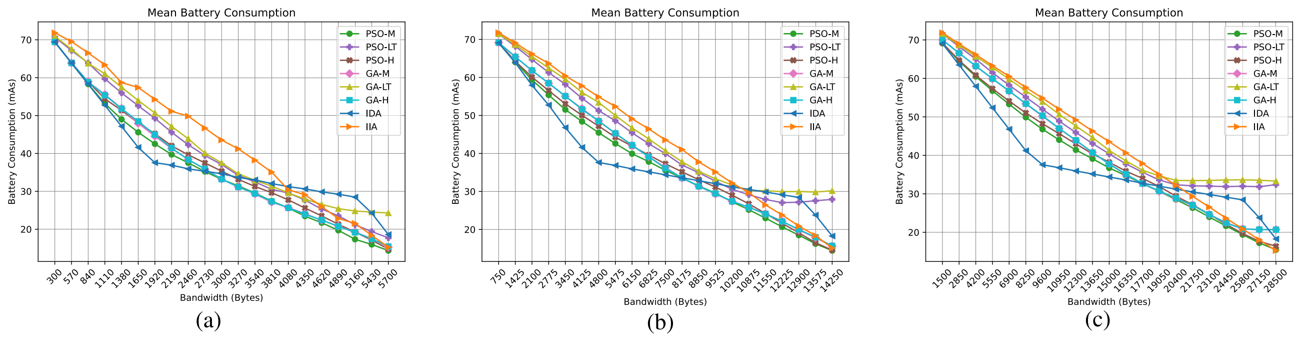


Fig. 7. Average consumption results for networks with varying available bandwidth. (a) 20 Nodes. (b) 50 Nodes. (c) 100 Nodes.

that would be obtained without offloading (all nodes at level 1), we obtain a relative percentage difference ranging from 4.24% to 134.5%, respectively. These results confirm that computational offloading has the potential to be a strategic method to decrease battery consumption in WSNs.

The IDA algorithm outperforms the rest when the bandwidth is lower than 8175 bytes in the case of networks with 50 nodes. This value represents a bandwidth in which 55% of the nodes could be in level 5. The same result is obtained for 20- and 100-node networks, indicating that IDA's bandwidth allocation method works best when bandwidth is limited but becomes less effective as bandwidth increases.

The cause of the lower performance of IIA for lower bandwidths relies on its bandwidth allocation method. In the solutions proposed by this algorithm, there is a higher probability of having nodes at level 1, which is the one with the highest consumption. For example, focusing on the results for a 50-node network with a bandwidth of 4800 bytes, IIA proposes a solution where two nodes occupy 200 bytes and the rest 90 bytes at each DC, giving an average consumption of 37.62 mAs not using the total available bandwidth. On the other hand, IIA proposes that 16 nodes occupy the maximum bandwidth, 300 bytes, and the rest compute the whole task locally leading to an average consumption of 58.872 mAs. This means that IDA's consumption is 37% lower than IIA's even though the latter uses the entire available bandwidth. Therefore, achieving a fair bandwidth distribution is more important than prioritizing the usage of the whole bandwidth. The fact that the consumption of IIA improves with increasing bandwidth, thereby making the distribution performed by this algorithm more homogeneous, corroborates this observation.

In the case of the metaheuristic algorithms, it is noticeable that, except for GA-LT and PSO-LT, they follow a decreasing trend very close to linearity and tend to improve the IDA algorithm as the bandwidth increases.

To appreciate the differences more precisely, the graph in Fig. 8 is generated. It represents the division of the average consumption of IDA by PSO-H (the best metaheuristic) of the 50-network results. A value lower than one indicates that the consumption generated by IDA is greater than that of PSO-H, and when it is greater than one, the opposite is true. Focusing on the values, the minimum point of this graph (0.797) indicates that IDA reduces the consumption generated by IDA by 22.6% compared to PSO-H for this bandwidth, while the maximum shows that PSO-H achieves consumption

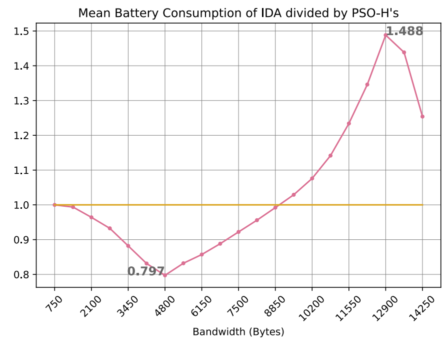


Fig. 8. 50-Node network: mean battery consumption of IDA divided by PSO-H.

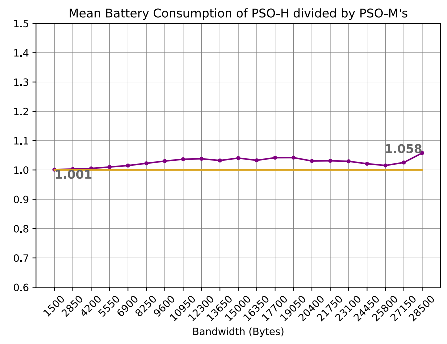


Fig. 9. 50-Node network: mean battery consumption of PSO-H divided by PSO-M.

39% lower than IDA when the bandwidth is greater, which is a considerable improvement. The same process has been followed for the networks with 20 and 100 nodes and the results are practically identical.

GA-LT and PSO-LT, the algorithms whose fitness is the network lifetime, lose linearity as the number of nodes and bandwidth increases and stagnates above a certain bandwidth. Raising the number of nodes adds complexity to the solutions, and higher bandwidths expand the search space, making the exploration of the solution more complex and causing the algorithms to lose efficiency. Consequently, they get stuck in a local-optimal solution and begin to plateau. The primary factor contributing to the stagnation of local optima in evolutionary algorithms is parameterization. Since the parameters are constant for all the algorithms, it can be inferred that the GA-LT and PSO-LT algorithms demonstrate diminished scalability compared to others. Furthermore, these findings can serve as a reference for determining when to modify the algorithm parameters. For instance, in a network consisting of 50 nodes,

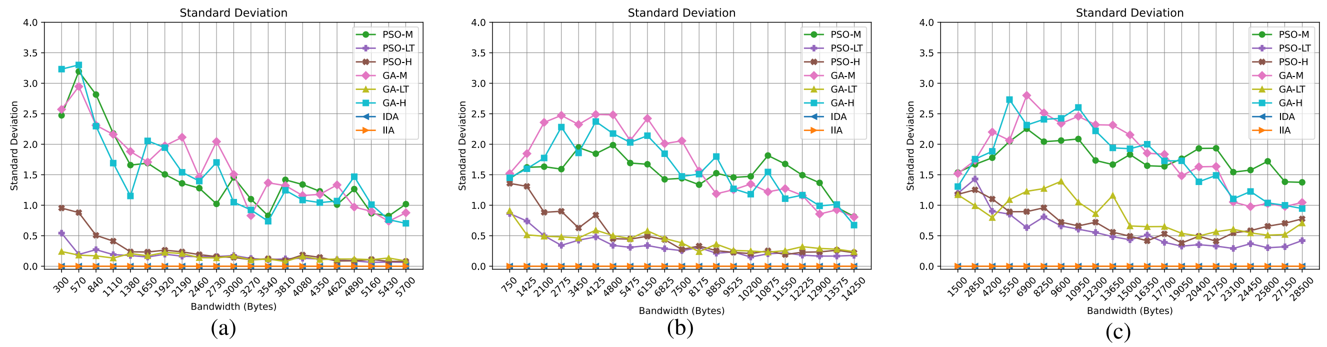


Fig. 10. Standard deviation results for networks with varying available bandwidth. (a) 20 Nodes. (b) 50 Nodes. (c) 100 Nodes.

adjusting the parameters would not be necessary if it is not expected that the BS could share more than 10 875 bytes.

The primary objective of metaheuristic algorithms incorporating fitness lifetime is not explicitly geared toward reducing the average consumption. Consequently, the mean consumption outcomes for these algorithms are relatively inferior compared to alternative metaheuristics. Nonetheless, within this subset, PSO-LT exhibits superior performance than GA-LT.

Similarly, metaheuristic algorithms incorporating fitness mean consumption exhibit analogous trends. Within this subset, it is noticeable that PSO-M achieves a lower consumption profile compared to GA-M and GA-H. This observation suggests that PSO algorithms have a greater resilience against becoming entrapped in local suboptimal solutions.

Finally, the results obtained for the PSO-H are worth highlighting. The graph shows that the PSO-H line is very close to the GA-M, GA-H, and PSO-M lines. The graphs in Fig. 9 are obtained using the same comparison procedure as IDA and PSO-H. The results indicate that PSO-H, which has a weighted fitness function in which the average consumption weights 60%, obtains values that do not exceed 5.6% of those algorithms whose fitness is represented exclusively by the average consumption of the network. The graphs for 20- and 50-node networks are very similar with even lower maximum values.

B. Standard Deviation Results

Fig. 10 illustrates the standard deviation results. Across the three networks, each graph exhibits three distinguishable ranges: iterative algorithms in the lower range, metaheuristics with medium consumption fitness in the upper range, and metaheuristics employing the network's lifetime as fitness, along with PSO-H, in the middle range.

The iterative algorithms have zero standard deviation, which is an expected result since the execution method of these algorithms forces this zero deviation.

The metaheuristics with the fitness set as average consumption have the highest standard deviation. GA-M and PSO-M do not include an explicit selection logic to maximize the network's lifetime. However, GA-H does when selecting the elite individual, so a lower standard deviation is expected when compared to GA-M, which is to some extent true for most bandwidths, although the difference is relatively small. On the other hand, the PSO-M has the lowest deviations in this range. They follow a trend where the standard deviation increases in the first few bandwidths and then decreases relatively steadily. The fact that level 1 occupies 0 bytes of bandwidth explains why at first, with low bandwidth, many nodes are limited to that level, generating a low standard deviation. As bandwidth increases, some nodes may move up to higher levels, but still many remain at level 1, increasing variability and standard deviation.

The transition in the slope around bandwidth 4500 in networks of 50 and 9000 in networks of 100 coincides with the point where all nodes can occupy level 2, which requires 90 bytes of bandwidth. This explains the change in the trend, as there is now sufficient capacity for all nodes to access level 2, increasing the homogeneity of the solutions.

As bandwidth continues to increase, nodes have more space to spread out over higher levels, which decreases dispersion and contributes to more homogeneous solutions in terms of average consumption.

Finally, metaheuristic algorithms that calculate their fitness as the network's lifetime occupy an intermediate range close to that of the iterative ones. This indicates that the algorithms meet their objective satisfactorily. The most outstanding result is that of PSO-H, which weights the average lifetime of the network by 0.4 and achieves values very close to PSO-LT and GA-LT.

While examining the results across different simulated networks, a noticeable trend emerges: as the number of nodes increases, there is a corresponding rise in the obtained values. This apparent escalation could be attributed to the surge in complexity during the search for solutions. The substantial increase in the number of nodes, coupled with the unchanged algorithm parameters, heightens this complexity. It is essential to highlight that this decision to maintain fixed parameters aims to emulate a realistic scenario where nodes are incrementally added to an existing network. Consequently, these findings underline a critical consideration: to ensure sustained efficiency, there should be an adaptation of algorithm parameters when undergoing a substantial expansion in the number of nodes.

C. Time Complexity Results

Time complexity is a crucial variable when considering the feasibility of implementing an algorithm. This factor defines the working cycle along with the maximum number of tasks the network would be capable of executing. The results

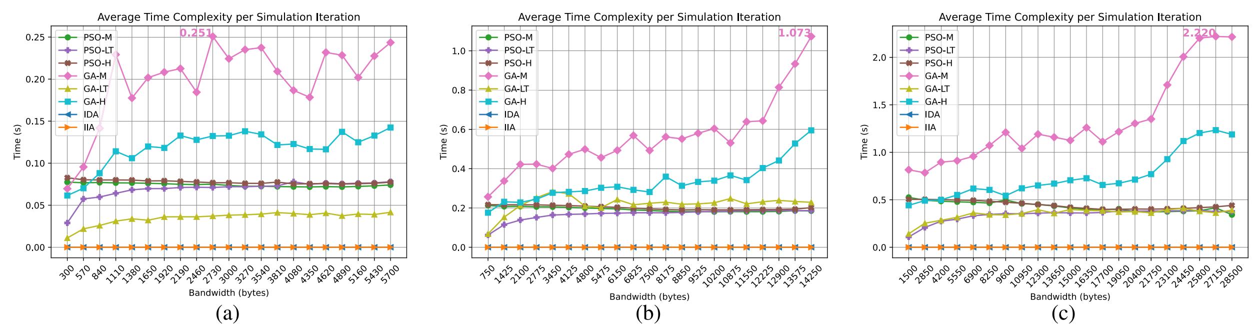


Fig. 11. Time complexity results for networks with varying available bandwidth. (a) 20 Nodes. (b) 50 Nodes. (c) 100 Nodes.

presented in Fig. 11 illustrate the average time the proposed algorithms have required to reach the optimal solution at each DC. The results will be discussed comparatively since runtime is closely related to the characteristics of the BS's processor. Additionally, in the case of metaheuristics, the convergence to optimal solutions presents some randomness. This inherent variability in metaheuristics can influence runtime, as the search process does not follow a deterministic path.

In light of the lower complexity of iterative algorithms compared to metaheuristics, the results indicate a higher temporal efficiency of the former. In the domain of metaheuristic algorithms, their complexity increases with the bandwidth across all three networks, this increment varies considerably from one algorithm to another. For instance, in simulations conducted on 50-node networks, the time complexity of GA-M increases by 318% when the bandwidth escalates from 750 to 14 250 bytes. In contrast, the PSO-H algorithm maintains a stable value throughout the bandwidths and experiences an increase of 14%.

To analyze the effect of increasing the number of nodes in the algorithms, if we compare the extreme case within a network, that is, the algorithms working with the maximum bandwidth tested, we observe that the execution time increases by a factor of 9.4 when increasing the number of nodes from 20 to 100, with the GA-LT algorithm experiencing the greatest increase among all the algorithms. On the contrary, in the case of PSO-M, it has the smallest increase with a value of 4.6. In general terms, it is observed that GA algorithms increase their time complexity to a greater extent than PSO algorithms. This may be related to the fact that there are fewer fluctuations in the values of these algorithms compared to the GAs.

The constancy in the average runtime of the PSO algorithms suggests the possibility that they do not reach the termination condition, resulting in all experiments completing the same number of generations. This phenomenon is closely related to the adaptations made in the search space to ensure compliance with the bandwidth restriction. However, as detailed in Section VI-A, it has been found that, despite these adaptations, PSO algorithms manage to obtain solutions that generate lower consumption compared to GAs.

D. Selection Guidelines

The selection of the most suitable algorithm for a given application depends on the characteristics of the network and the main objective to be achieved with the application of task offloading. Among the proposed algorithms, some outperform others in certain aspects at the cost of lower performance in others. To visualize the strengths and weaknesses of each of the algorithms and thus facilitate their selection for each application, a comparative evaluation based on a scoring system is carried out.

The assessment of the algorithms involves the evaluation of five key aspects: their suitability in dynamic network contexts, average consumption, standard deviation, execution time, and adaptability to an increase in the number of network nodes. Each of these aspects is assigned a score on a scale from one to eight. Table V presents the results of this assessment, where the following criteria are applied.

1) The scoring process for algorithms in terms of average consumption, standard deviation, and execution time follows a systematic ranking system. First, the results of the algorithms for each bandwidth of a given network are arranged from best to worst. Initial scores are then assigned in descending order, ranging from 8 to 1 points. In instances of tied performance among algorithms—where multiple algorithms exhibit very similar results—the tied group collectively receives the highest score within that group. The subsequent algorithm in the original ranking retains its initially assigned score. This scoring procedure is iterated for each considered bandwidth, and the final score for each algorithm across a specific network is determined by calculating the average.
2) Algorithms that consider the state of the batteries to generate solutions were given a score of 8, indicating their suitability for use in dynamic network contexts. Conversely, those that do not take this aspect into account received a score of 1 for this criterion.
3) Scalability has been scored so that the algorithms that maintain their tendencies throughout the different networks obtain eight points, PSO-LT and GA-LT are given four points since they begin to stagnate on 50-node networks and GA-M and GA-H received seven points because they stagnate on the final bandwidths on the 100-node network.

The results shown in Table V depict the average scores derived from three networks. The total count of the points obtained by each algorithm is carried out in a way that ensures equal weight is given to each of the evaluated aspects. The algorithms that have obtained the highest scores within each category are highlighted in the table. These results underline

TABLE V
ALGORITHMS SCORE

	IDA	IIA	GA-M	GA-LT	GA-H	PSO-M	PSO-LT	PSO-H
Dynamic network	1	1	8	8	8	8	8	8
Mean Consumption	5.7	2.3	6.6	2.3	6.6	7.6	3	6.3
Standard deviation	8	8	1.8	5.2	2	2.5	5.9	5.2
Execution time	8	8	1	3.9	2.6	5.1	5.9	4.7
Scalability	8	8	7	4	7	8	4	8
Total	**30.7**	27.3	24.4	23.4	**26.2**	31.2	26.8	**32.2**

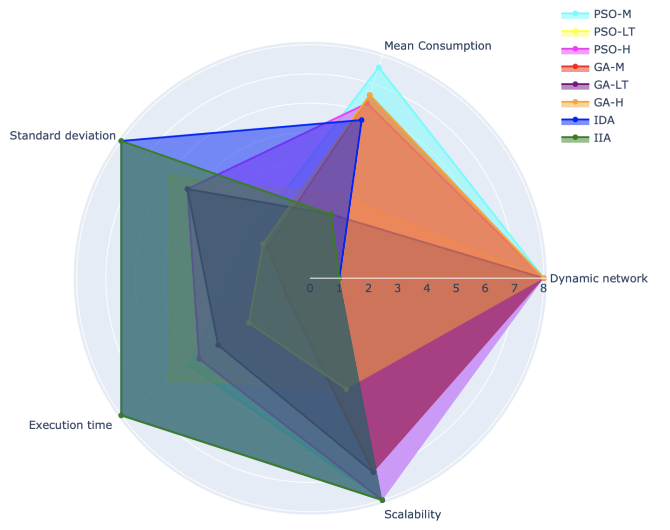


Fig. 12. Selection guidance chart of the task-offloading algorithms.

the overall outperformance of the multiobjective implementation of the PSO-H algorithms.

Furthermore, in addition to being used to evaluate the algorithms, observations of algorithm performance under different bandwidth conditions provide valuable insight into the predeployment sizing of the network and BS. An understanding of the behavior of the algorithms with different bandwidths can help determine the optimal number of nodes based on the bandwidth range of the selected edge device to ensure the network's longevity. Similarly, knowing the range in which the network's node count will fluctuate can establish the bandwidth range needed to ensure the durability of the network. In turn, this would guide the selection of an appropriate edge device capable of providing these characteristics.

VII. Conclusion and Further Research

This work examines the benefits of task offloading in resource-constrained WSNs. The primary goal is to reduce node battery consumption by partially or completely transferring computation tasks to a BS. Empirical results indicate that even the least efficient offloading strategy demonstrates a significant reduction in energy consumption, affirming the overall efficacy of task offloading. Evaluation of the different algorithms emphasizes the role of selecting strategies tailored to specific application objectives. However, when considering all the evaluated aspects, the multiobjective approach of the PSO-H algorithm is particularly noteworthy. This study provides a valuable roadmap for implementing offloading in WSNs, with future research poised to refine and adapt these strategies in response to the dynamic landscape of technological demands.

The study's contributions extend beyond demonstrating the efficacy of task offloading on WSN and underscore the importance of strategic selection aligned with specific application goals. The multiobjective approach of the PSO-H algorithm emerges as a notable contribution, providing a promising direction for implementing offloading strategies in WSNs.

As discussed in [43], artificial intelligence (AI) and machine learning (ML) methods have the potential to play an essential role in decision-making in environments where task offloading is applied. In particular, reinforcement learning, a branch of AI, could be a powerful tool in this area. These methods could be very useful to consolidate and analyze the results obtained in our current research. Moreover, they could pave the way for the development of an intelligent tool capable of selecting the most suitable computational offloading strategy according to the provided network parameters. This perspective opens up new possibilities to automate and optimize decision-making in the context of task offloading in WSNs, thus improving their efficiency and performance.

References

[1] J. Wang, M. K. Lim, C. Wang, and M.-L. Tseng, "The evolution of the Internet of Things (IoT) over the past 20 years," *Comput. Ind. Eng.*, vol. 155, May 2021, Art. no. 107174.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] J. Portilla, G. Mujica, J.-S. Lee, and T. Riesgo, "The extreme edge at the bottom of the Internet of Things: A review," *IEEE Sensors J.*, vol. 19, no. 9, pp. 3179–3190, May 2019.

[4] C. M. Fernández, M. D. Rodríguez, and B. R. Mu noz, "An edge computing architecture in the Internet of Things," in *Proc. IEEE 21st Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2018, pp. 99–102.

[5] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[6] L. Wang, H. Shao, J. Li, X. Wen, and Z. Lu, "Optimal multi-user computation offloading strategy for wireless powered sensor networks," *IEEE Access*, vol. 8, pp. 35150–35160, 2020.

[7] A. Arora et al., "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Comput. Netw.*, vol. 46, no. 5, pp. 605–634, Dec. 2004.

[8] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[9] P. Merino, G. Mujica, J. Señor, and J. Portilla, "A modular IoT hardware platform for distributed and secured extreme edge computing," *Electronics*, vol. 9, no. 3, p. 538, 2020.

[10] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2017.

[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[12] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[13] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[14] D. Xu et al., "A survey of opportunistic offloading," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2198–2236, 3rd Quart., 2018.

[15] M. A. Khan, "A survey of computation offloading strategies for performance improvement of applications running on mobile devices," *J. Netw. Comput. Appl.*, vol. 56, pp. 28–40, Oct. 2015.

[16] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. 13th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2012, pp. 145–154.

[17] M. H. U. Rehman, S. L. Chee, T. Y. Wah, A. Iqbal, and P. P. Jayaraman, "Opportunistic computation offloading in mobile edge cloud computing environments," in *Proc. 17th IEEE Int. Conf. Mobile Data Manage. (MDM)*, vol. 1, Jun. 2016, pp. 208–213.

[18] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 7–12.

[19] F. Xu, H. Ye, F. Yang, and C. Zhao, "Software defined mission-critical wireless sensor network: Architecture and edge offloading strategy," *IEEE Access*, vol. 7, pp. 10383–10391, 2019.

[20] P. Rong and M. Pedram, "Extending the lifetime of a network of battery-powered mobile devices by remote processing: A Markovian decision-based approach," in *Proc. 40th Design Autom. Conf.*, 2003, pp. 906–911.

[21] Z. Li and Q. Zhu, "Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing," *Information*, vol. 11, no. 2, p. 83, 2020.

[22] S. Fu, C. Ding, and P. Jiang, "Computational offloading of service workflow in mobile edge computing," *Information*, vol. 13, no. 7, p. 348, Jul. 2022.

[23] H. R. Lourenco and D. Serra, "Adaptive search heuristics for the generalized assignment problem," *Mathware Soft Comput.*, vol. 9, nos. 2–3, pp. 1–3, 2002.

[24] T. Zheng, J. Wan, J. Zhang, C. Jiang, and G. Jia, "A survey of computation offloading in edge computing," in *Proc. Int. Conf. Comput., Inf. Telecommun. Syst. (CITS)*, Oct. 2020, pp. 1–6.

[25] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *J. Syst. Archit.*, vol. 118, Sep. 2021, Art. no. 102225.

[26] Q. Shi, X. Wang, W. Chen, and K. Hu, "Optimal sensor placement method considering the importance of structural performance degradation for the allowable loadings for damage identification," *Appl. Math. Model.*, vol. 86, pp. 384–403, Oct. 2020.

[27] Q. Shi, K. Hu, L. Wang, and X. Wang, "Uncertain identification method of structural damage for beam-like structures based on strain modes with noises," *Appl. Math. Comput.*, vol. 390, Feb. 2021, Art. no. 125682.

[28] C. Yang and H. Ouyang, "A novel load-dependent sensor placement method for model updating based on time-dependent reliability optimization considering multi-source uncertainties," *Mech. Syst. Signal Process.*, vol. 165, Feb. 2022, Art. no. 108386.

[29] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Metaheuristic algorithms: A comprehensive review," in *Computational Intelligence for Multimedia Big Data on the Cloud With Engineering Applications*. New York, NY, USA: Academic, 2018, pp. 185–231.

[30] H. Jh, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: MIT Press, 1975.

[31] A. A. Al-habob, O. A. Dobre, and A. Garcia Armada, "Sequential task scheduling for mobile edge computing using genetic algorithm," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6.

[32] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artif. Intell. Rev.*, vol. 12, no. 4, pp. 265–319, 1998.

[33] K. Jebari and M. Madiafi, "Selection methods for genetic algorithms," *Int. J. Emerg. Sci.*, vol. 3, no. 4, pp. 333–344, 2013.

[34] C. W. Ahn and R. S. Ramakrishna, "Elitism-based compact genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 367–385, Aug. 2003.

[35] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN')*, vol. 4, 1995, pp. 1942–1948.

[36] J. Blondin. (2009). *Particle Swarm Optimization: A Tutorial*. [Online]. Available: http://cs.armstrong.edu/saad/csci8100/psotutorial.pdf

[37] W. Chuanjun, W. Ling, and R. Xuejing, "General particle swarm optimization algorithm," in *Proc. IEEE 2nd Int. Conf. Electr. Eng., Big Data Algorithms (EEBDA)*, Feb. 2023, pp. 1204–1208.

[38] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.

[39] Y. Cui, Z. Geng, Q. Zhu, and Y. Han, "Multi-objective optimization methods and application in energy saving," *Energy*, vol. 125, pp. 681–704, Apr. 2017.

[40] F. Wu, X. Fu, P. Lang, J. Dong, Z. Cui, and X. Gao, "Cognitive radar waveform design based on multi-objective optimization criteria," in *Proc. 7th Int. Conf. Signal Image Process. (ICSIP)*, Jul. 2022, pp. 172–176.

[41] M. S. Gibbs, H. R. Maier, G. C. Dandy, and J. B. Nixon, "Minimum number of generations required for convergence of genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jul. 2006, pp. 565–572.

[42] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Population size in particle swarm optimization," *Swarm Evol. Comput.*, vol. 58, Nov. 2020, Art. no. 100718.

[43] S. Iftikhar et al., "AI-based fog and edge computing: A systematic review, taxonomy and future directions," *Internet Things*, vol. 21, Apr. 2023, Art. no. 100674.

Paula González received the B.Sc. degree in industrial technologies engineering from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 2023, where she is currently pursuing the M.Sc. degree in industrial electronics engineering.

She has been collaborating for one year with the Center of Industrial Electronics Researching Energy-Saving Strategies in the Internet of Things (IoT). Her research interests focus on studying novel optimization methods for task offloading in wireless sensor networks and IoT.

Gabriel Mujica (Member, IEEE) received the Ph.D. degree in industrial electronics engineering from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 2017.

He is an Associate Professor and a Research Member at the Center of Industrial Electronics, Universidad Politécnica de Madrid (CEI-UPM), where he is mainly involved in the area of Internet of Things (IoT), networked embedded systems, and wireless sensor networks (WSNs). He has participated in different national and European research projects (including Horizon 2020 Projects), related to the development and optimization of WSN, as well as the integration of heterogeneous IoT edge hardware, software, and communication technologies for wireless distributed systems, with a particular focus on the performance evaluation and optimization of sensor platforms under real deployment contexts. He has collaborated in the organization of research tutorials and seminars and as a reviewer and guest editor in international conferences and indexed journals. Moreover, his visiting research stay at the Trinity College Dublin, Dublin, Ireland, strengthened the vision and applicability of IoT technologies for smart and sustainable cities, leveraging collaborations in the area of distributed systems within such contexts. He has authored several contributions in high-impact conferences and journals. Currently, his main research interests are related to multihop distributed networks for the extreme edge of IoT, hardware-software co-design and communication protocols for IoT embedded systems in smart urban and industrial application scenarios.

Dr. Mujica received the International Distinction and Outstanding Doctorate Award for his Ph.D. degree.

Jorge Portilla (Senior Member, IEEE) received the M.Sc. degree in physics from the Universidad Complutense de Madrid, Madrid, Spain, in 2003, and the Ph.D. degree in electronic engineering from the Universidad Politécnica de Madrid (UPM), Madrid, in 2010.

He was a Visiting Researcher with the Industrial Technology Research Institute, Hsinchu, Taiwan, in 2008, and also with the National Taipei University of Technology (Taipei Tech), Taipei, Taiwan, in 2018, working on wireless sensor networks hardware platforms and network clustering techniques. He is currently an Associate Professor with the Universidad Politécnica de Madrid. He has participated in more than 30 funded research projects, including European Union FP7 and H2020 projects, and Spain Government funded projects, as well as private industry funded projects, mainly related to wireless sensor networks and the Internet of Things. He has numerous publications in prestigious international conferences as well as in journals with impact factor. He carried out his research activity within the Centro de Electrónica Industrial, belonging to the UPM. His research interests are focused on wireless sensor networks, the Internet of Things, digital embedded systems, and reconfigurable FPGA-based embedded systems.