future internet

MDPI

Article

Improved Dragonfly Optimization Algorithm for Detecting IoT Outlier Sensors

Maytham N. Meqdad [1], Seifedine Kadry [2,3,4] and Hafiz Tayyab Rauf [5,*]

[1] Computer Techniques Engineering Department, Al-Mustaqbal University College, Hillah 51001, Iraq
[2] Department of Applied Data Science, Noroff University College, 4612 Kristiansand, Norway
[3] Department of Electrical and Computer Engineering, Lebanese American University, Byblos 1102, Lebanon
[4] Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, Ajman 20550, United Arab Emirates
[5] Centre for Smart Systems, AI and Cybersecurity, Staffordshire University, Stoke-on-Trent ST4 2DE, UK
* Correspondence: h.rauf4@bradford.ac.uk

Abstract: Things receive digital intelligence by being connected to the Internet and by adding sensors. With the use of real-time data and this intelligence, things may communicate with one another autonomously. The environment surrounding us will become more intelligent and reactive, merging the digital and physical worlds thanks to the Internet of things (IoT). In this paper, an optimal methodology has been proposed for distinguishing outlier sensors of the Internet of things based on a developed design of a dragonfly optimization technique. Here, a modified structure of the dragonfly optimization algorithm is utilized for optimal area coverage and energy consumption reduction. This paper uses four parameters to evaluate its efficiency: the minimum number of nodes in the coverage area, the lifetime of the network, including the time interval from the start of the first node to the shutdown time of the first node, and the network power. The results of the suggested method are compared with those of some other published methods. The results show that by increasing the number of steps, the energy of the live nodes will eventually run out and turn off. In the LEACH method, after 350 steps, the RED-LEACH method, after 750 steps, and the GSA-based method, after 915 steps, the nodes start shutting down, which occurs after 1227 steps for the proposed method. This means that the nodes are turned off later. Simulations indicate that the suggested method achieves better results than the other examined techniques according to the provided performance parameters.

Keywords: Internet of things; sensor detection; improved dragonfly optimization algorithm

Citation: Meqdad, M.N.; Kadry, S.; Rauf, H.T. Improved Dragonfly Optimization Algorithm for Detecting IoT Outlier Sensors. *Future Internet* **2022**, *14*, 297. https://doi.org/10.3390/fi14100297

Academic Editors: Sylvain Kubler and Jérémy Robert

Received: 28 August 2022
Accepted: 10 October 2022
Published: 17 October 2022

1. Introduction

The Internet is slowly moving and migrating from the Internet of people to the Internet of things. Internet of things (IoT) devices can become pervasive and enable context-aware and environmental intelligence [1,2]. However, the ability to interact between heterogeneous Internet objects, mobile handheld devices, and wireless sensors faces severe complications [3]. A network of sensors contains self-organizing networks made up of several nodes dispersed across an area that gather the necessary data and transmit them to a base station node [4]. Sensors are crucial parts of intelligent things. One or more sensors are necessary for all IoT applications to gather environmental data. Receiving information, which is essential to the IoT, is only feasible with sensing devices.

Internet of things sensors are primarily tiny, low-power, and low-cost, limited by features such as their ease of deployment and battery capacity [5]. In this research, the focus is on integrating sensor networks and the Internet of things, as well as detecting outlier sensors [6]. However, the development of the Internet of things faces several obstacles. These difficulties are social and professional. These obstacles need to be cleared away to guarantee the adoption and penetration of the Internet of things.

The location and proper communication of sensors are essential in sensor-based computer networks to perform optimal tasks. In addition, location information is helpful in geographic routing and joint signal processing [7]. Using different methods, researchers try to choose the best location for sensors so that service delivery in sensor-based networks can be adequately carried out [8].

One of the suitable approaches to removing network outlier sensors is the use of location knowledge. However, in the meantime, a sensor or sensors with incorrect placement may challenge the network [9].

Such sensors will be called outlier sensors in this article. Location is the best place to establish service providers' facilities, and they are also trying to reduce lost demand by rationally allocating the demand centers to them. Sensor positioning is also a subset of these issues [10].

The placement of sensors in sensor-based networks seems impractical due to the nature of such networks; in addition to many sensors, their position is inaccessible in some applications [11]. Although the global locating system may be utilized to determine the location of sensors, its use is not always possible, in addition to it being expensive [12]. Therefore, alternative methods should be used.

The Internet of things is a subset of computer networks that have recently gained popularity. In this network, a set of sensors is placed in a defined workspace, providing services by establishing communication [13]. In the meantime, a sensor with a lousy placement may challenge the network.

Technological advances in low-power integrated circuits have made it possible to build low-cost as well as small-sized sensor nodes and create wireless sensor networks by connecting them. Sensor nodes can sense, process, and send data [14]. The significant discrepancy between communication networks and these sensor networks is their data-centric nature and minimal energy as well as processing resources.

In a sensor network, many small nodes autonomously monitor and interact with the environment. These sensors take information from the environment and transfer it to a data collection center [15]. One of the characteristics of these sensors is independence and operation without human intervention. Sensors can be used in small spaces due to their small size and limited memory, processing power, and battery. Since there is no base station in wireless sensor networks and their radio frequency is low, the dispersion of nodes should be such that the nodes are in each other's radio coverage and deliver the sensed data to the destination from the nearest route while thoroughly covering the area. Usually, due to the geographical location of these types of networks in dangerous environments and the remote locations of some nodes, it is impossible to recharge or use them properly [16]. Therefore, choosing the correct location of sensors and removing extraneous sensors increase a system's efficiency and cause it to consume less energy. This paper has tried to propose an approach to detect outlier sensors of the Internet of things by using a genetic algorithm. In this article, after reviewing the Internet of things and its challenges, a modified design of the dragonfly optimization algorithm is introduced, and the suggested technique is described. In the following, after introducing the simulation environment, simulation parameters, and evaluation criteria, the research findings are presented. The main contributions of this paper can be highlighted as follows:

- Recommends a modified dragonfly optimization algorithm structure for better area coverage and reduced energy use.
- Utilizes the minimal number of nodes in the coverage area, the lifetime of the network, which includes the period from the commencement of the first node to the time of the first node's shutdown, and the network power to assess efficiency.
- Makes comparisons with some other approaches that have been published.

The structure of the paper is as follows: In Section 2, a literature review concerning the method is explained. Section 3 describes the method of designing for providing the improved version of the dragon fly optimizer. In Section 4, the optimal coverage

methodology for the problem based on the improved dragon fly optimization algorithm is explained. Section 5 defines the simulation results, and the paper is concluded in Section 6.

2. Literature Review

In this section, some related papers from the field of the detection of outlier sensors and the positioning of sensors in the field of the Internet of things have been studied [16].

Deng et al. [17] presented a tensor Tucker-based OCSTuM with a GA for the intellectual outlier detection of big sensor data. This technique takes one-class SVMs and expands them to tensor space. The OCSTuM and GA-OCSTuM were autonomous outlier detection methods for large amounts of sensor data. They preserved data structure while boosting the effectiveness and precision of outlier detection. Experiments on real-world datasets showed that their suggested strategy enhanced the accurateness of anomaly identification while preserving the inherent structure of massive sensor data.

Titouna et al. proposed a distributed outlier detection system for WSNs [18]. They introduced the DODS (distributed outlier detection scheme), where various types of data were examined and outliers were recognized locally by all nodes by a collection of classifiers, such that neither knowledge of nearby neighbors nor connections between nodes were needed. These features made the suggested approach effective and scalable in terms of both energy usage and communication costs. The suggested scheme's functions were evaluated by extensive simulations utilizing real-world data gathered from the Intel Berkeley Research Lab. The collected findings proved the suggested scheme's efficacy in comparison to the examined algorithms.

Ferrer-Cid et al. proposed outlier detection based on the Volterra graph method for air pollution sensor networks [19]. To properly assess the outliers of the sensors that comprise a network, we suggest the VGOD (Volterra graph-based outlier detection) method, which detects and localizes anomalous indicators in air pollution sensor networks using a graph educated from data and a Volterra-like graph signal reforming model. The suggested unsupervised process was compared to some other published works, including graph-based and non-graph-based ones, and shows advancements in both the recognition and placement of outlier measured data, allowing irregular measurements to be corrected and misbehaving sensors to be supplanted.

Dwivedi et al. [20] presented a study on machine learning methods for outlier detection in WSNs. In this study, machine-learning-based strategies for outlier detection were examined, with a Bayesian network appearing to be superior to other methods. In a WSN, the Bayesian classification technique may be utilized to calculate the conditional reliance of the accessible nodes. Based on the explanations in the paper, the technique can also compute the value of missing data.

Gil et al. introduced outlier sensor detection in WSNs [21]. They attempted to close this gap by providing an experimental assessment of two cutting-edge online detection algorithms. The first technique depends on LS-SVM and a sliding-window-based learning algorithm, whereas the next technique is based on PCA and robust orthonormal projection estimation subspace tracking with rank-1 adjustment. The efficiency and applicability of the approaches were assessed by a testbed and produced nonstationary time series comprising a benchmark three-tank system and a WSN in which organized algorithms were applied by a multiagent outline. In the following Table 1, a brief explanation of the literature is given.

Recently, utilizing optimization algorithms, especially metaheuristics, for the purpose of the detection of IoT outlier sensors has been enhanced. The results show that these techniques have substantially better outcomes than the classic optimization algorithm. However, because of the no free lunch theory, there is no optimization algorithm can be the best when compared to others. This being the case, in this paper we utilized an algorithm to solve this issue.

Table 1. A brief explanation of the literature.

Author	#Ref.	Date	Findings
Deng et al.	[17]	2018	Tensor Tucker-based OCSTuM with a GA for the intellectual outlier detection of big sensor data
Chafiq et al.	[18]	2019	Distributed outlier detection system for WSNs
Ferrer-Cid et al.	[19]	2022	Volterra graph method for the air pollution sensor networks
Dwivedi et al.	[20]	2018	Outlier detection in WSNs based on a Bayesian network
Gil et al.	[21]	2019	PCA and robust orthonormal projection estimation subspace tracking

3. Improved Dragonfly Optimizer

3.1. The Basic Dragonfly Optimizer

Five key components update the locations of living objects in the dragonflies' group movement, each of which is mathematically represented, considering the two groups of static and dynamic group movements. Calculating separation is performed as follows [22]:

$$S_i \sum_{j=1}^{N} y - y_j \tag{1}$$

where y depicts the dragonfly's current position, y_j explains the neighboring dragonfly location of j, and N influences the number of dragonfly communities [23]. The following is how the balance is calculated:

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \tag{2}$$

where V_j determines the dragonfly's acceleration in the vicinity of j. The connection is depicted below:

$$C_i = \frac{\sum_{j=1}^{N} y_j}{N} - y \tag{3}$$

The following model is used to simulate absorption into food:

$$F_i = y^+ - y^- \tag{4}$$

where X^+ denotes the basic nation of the food. The distance from the adversary is also specified here:

$$E_i = y^- + y^+ \tag{5}$$

Here, y^- designates the position of the adversary.

The DA optimization algorithm considers dragonfly performance to be a combination of these five types. Two variables, Δy and location, y, are explored to refresh the dragonflies' positions in the candidate solutions and their movement simulations [24].

This variable is characterized as a model for changing the one-dimensional location of the solution candidates (while this method can be widespread to higher dimensions), which equals the following:

$$\Delta y_{t+1} = (sS_i + aA_i + cC_i + eE_i + fF_i) \tag{6}$$

where s shows the separating coefficient, S_i determines the level of separation in candidates, i, a shows the optimum weight, A_i defines the dragonfly's level of balance, i, c shows the significance of continuity, C_i shows the quantity of cohesiveness in dragonflies, i, f is the feed parameter, f_i explains the dragonfly's source of nutrients, i, e determines the enemy

factor, E_i shows the location of the dragonfly's adversary, w is the inertia factor, and t is the quantity of iterations. The following equation was used to modify the position variables:

$$y_t = y_t + \Delta y_t \tag{7}$$

The utilization of five parameters, namely separation, equilibrium, continuous, feed, and opponent, may be carried out throughout the capability optimization procedure. To enhance the search for a metaheuristic technique, dragonfly neighbors are quite crucial. As a result, dragonflies must be a particular radius apart (circular 2D area, a circle in 3D space, and a multidimensional sphere in nD space).

This dragonfly is formed to develop the potential behavior of the algorithm. It must fly in an environment with no local reaction. The Levy approach was utilized in the construction of the DA optimization algorithm in this regard.

The position of the dragonflies has been updated as follows:

$$y_{t+1} = y_t + Levy(d) \times y_t \tag{8}$$

where t defines the value of iterations and d shows the position's dimension vector. The following formula can be employed to determine the $Levy(d)$ function:

$$Levy(d) = 0.01 \times \frac{r_1}{|r_2|^{\frac{1}{\beta}}} \times \sigma \tag{9}$$

where r_1 and r_2 define stochastic quantities between 0 and 1 and β one fixed number (generally equal to $1/5$), and the variable σ is equal to:

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \tag{10}$$

The equation for function $\Gamma(y)$ seems to be as follows:

$$\Gamma(X) = (y-1)! \tag{11}$$

The DA algorithm's optimization procedure starts by creating a random set to address the optimal solution. In fact, the step vectors and locations of the dragonflies were chosen at random depending on the higher and lower bounds.

In each iteration, the location of each dragonfly is changed [25]. The neighbor of candidates is determined based on choosing N candidates and calculating the distance measure to update the y and Δy vectors. Up to the stop condition, the position will be updated continuously.

3.2. Improved Dragonfly Optimization Technique

Through researching the literature [26,27], whereas the basic dragonfly optimization technique is effective in resolving a variety of issues, it occasionally fails to do so and produces local optimal outcomes or premature convergence [28,29].

Two modifications have been taken into consideration in this study to address this issue. It is clear from the basic dragonfly optimization technique that the two random parameters r_1 and r_2 occasionally cause the algorithm to converge too soon [30]. This paper utilizes chaos theory to address this problem [31]; therefore, the singer mapping chaos function is used [32,33]. To convert the random data into regular pseudo-random values is the duty of vocalist mapping in this case [34]. The new r_1 and r_2 are obtained using this approach in the manner described below:

$$r_1^{i+1} = 1.07\left(7.9r_1^i - 23.3\left(r_1^i\right)^2 + 28.7\left(r_1^i\right)^3 - 13.3\left(r_1^i\right)^4\right) \tag{12}$$

$$r_2^{i+1} = 1.07\left(7.9r_2^i - 23.3\left(r_2^i\right)^2 + 28.7\left(r_2^i\right)^3 - 13.3\left(r_2^i\right)^4\right) \quad (13)$$

The program also now offers a quasi-oppositional learning algorithm for its fresh solutions. This adjustment has been made to fix the algorithm's premature convergence issue.

The newly created candidates from the second iteration have been compared to their symmetrical values in the quasi-oppositional learning process, and the best candidate is picked as the next rival.

Concerning the t^{th} candidate and y_t in the domain of solutions with constraints between $[Lb, Ub]$, the following is how the potential equal value is determined:

$$\begin{gathered} \overline{y}_t = Lb_i + Ub_i - y_t \\ i = 1, 2, \ldots, d. \end{gathered} \quad (14)$$

where d shows the dimension.

Additionally, the quasi-opposite amount, $(\overline{\overline{y}}_t)$ for y_t is obtained using the following equation:

$$\overline{\overline{y}}_t = rand(y_t,\ 0.5 \times Lb_i + Ub_i) \quad (15)$$

4. Optimize Coverage with an Improved Dragonfly Optimization Algorithm

In this section, the method of using the suggested improved dragonfly optimization algorithm for solving the coverage problem in wireless sensor networks is performed. Each object is assumed to be equal to one sensor node in dragonfly relations, such that all of the sensor nodes and the sinks have a fixed location after being arranged. Nodes were homogeneous and with the same communication capacities of the range for sensing and capacities for data processing. The location of each sensor node is already recognized, and the well knows the location of all of the nodes.

The radio range of the sensors is assumed to be r_s. The proposed algorithm assumes the problem of point coverage in a certain environment. As a result, outlier sensors (sensors that are not in the coverage area) will be removed. The point of interest (POI) is the place where the event takes place. It is assumed that the event signal can always be generated in all POIs. Figure 1 shows sending events at star points by neighboring nodes to the sink.

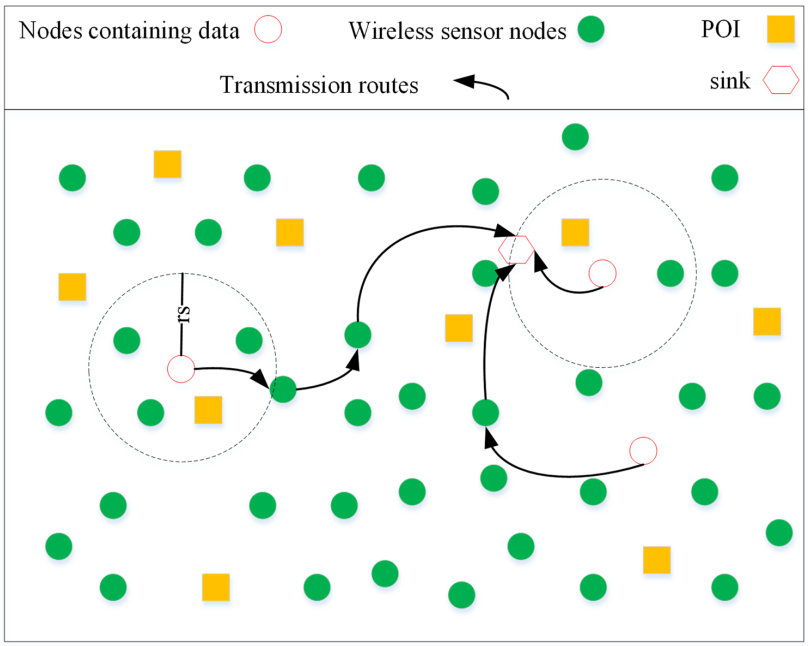

Figure 1. Events sending at star points with neighboring nodes to the sink.

As can be observed from Figure 1, each POI is enclosed by some sensor nodes. If the POI is in the interior of the range of the radio radius, r_s, of a certain sensor, the sensor node detects the event signal and sends the restrained data to the well in multistep.

The target environment, is a two-dimensional surface, $L_x \times L_y$ square meters.

Some sensor nodes in R have been described by $C = [c_1, c_2, \ldots, c_M]$, where $i \in [1, M]$, $c_i = [x_i, y_i, r_s]$ and M define the sensor nodes' quantity. The coordinates and radio beam of a sensor node are defined by $[x_i, y_i]$ and r_s, respectively. Some POIs distributed in region R are determined by $P = [p_1, p_2, \ldots, p_n]$, where p_k has been placed at $k \in [1, N]$ in $[x_k, y_k]$, and M is the number of POIs. The binary variable $LO_{i,j}$ in the following equation specifies whether c_i is able to cover P_k or not:

$$LO_{i,j} = \begin{cases} 1, & if \quad (x_i - x_j)^2 + (y_i - y_j)^2 < r_s^2 \\ 0, & Otherwise \end{cases} \tag{16}$$

If the distance between p_k and c_i provides small value then the r_s, then p_k will be covered by c_i. Therefore, the event signal made in the POI can be identified. Although p_k can be simultaneously covered by v sensor nodes, if a specific p_k has been sheltered by more than one sensor node, then the union of $LO_{1,j}$, $LO_{2,j}$, $\ldots$, $LO_{v,j}$ for p_j is calculated by the Boolean (binary) operator:

$$LO_{1,j} \vee LO_{2,j} \vee \ldots, LO_{v,j} = 1 - \overline{LO}_{1,j} \wedge \overline{LO}_{2,j} \wedge \ldots, \overline{LO}_{v,j} \tag{17}$$

where, $\overline{LO}_{i,j}$ is the Boolean inverse of $LO_{i,j}$. The operators $\vee$ and $\wedge$ represent the symbol of the Boolean union and intersection. The energy consumption model is shown in Figure 2. Therefore, most of the energy used in radio transmission is in the general operation of the sensor and access to the memory, so other energy-consuming matters are overlooked.

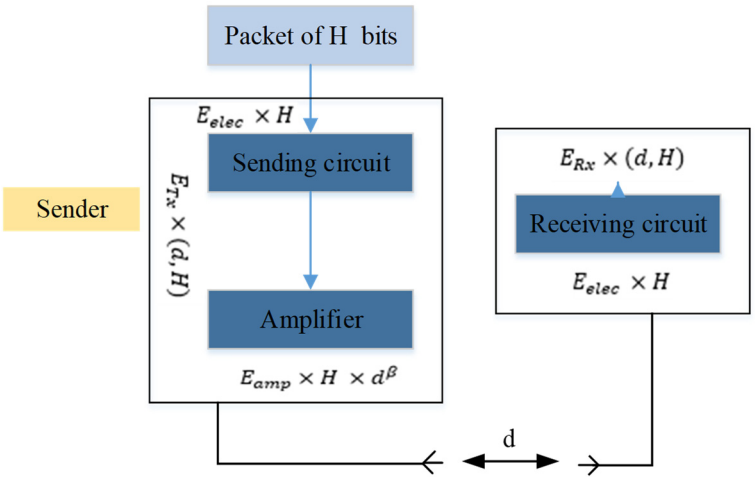

Figure 2. Model of energy consumption.

In the above Figure, E_{el} represents the energy lost in the transmission circuit or the receiving circuit per bit (in nanojoules). E_{Amp} indicates the energy consumed by the power amplifier per bit and β is the power of the path. The value of β is usually taken as 2. As a result, when an H-bit packet is sent to the receiver, the total energy consumed is calculated as follows:

$$E_{Tx}(d, H) = E_{el} \times H + E_{Amp} \times H \times d^{\beta} \tag{18}$$

$$E_{Rx}(d, H) = E_{el} \times H \tag{19}$$

where E_{Tx} and E_{Rx} represent, in turn, the overall energy used to send and receive an H-bit packet.

Since all of the sensor nodes are supposed to be homogeneous, their initial energy is the same. However, the energy of the sink is high and is not included in this energy consumption model. The set covering problem (SCP) is an NP-hard problem [35].

In this article, this concept is implemented on the node scheduling optimization, which contains the issue of coverage management along with energy efficiency. In fact, the SCP is the problem of finding some nodes with minimal costs, where desired points are enclosed by no less than one node. Therefore, the set covering problem can be defined as follows:

$$\min \sum_{i} g_i \cdot x_i, \quad i \in [1, M] \tag{20}$$

Such that:

$$\begin{gathered} \sum_{i} LO_{i,j} \cdot x_i \geq 1, \quad k \in [1, N] \\ x_i \in (0,\ 1),\ i \in [1, M] \end{gathered} \tag{21}$$

where, g_i is the activation cost of each node and $x_i = 1$ means activation and $x_i = 0$ means inactive (for outlier sensors). The constraint of the equation above states that every p_k is enclosed by no less than one node. After the network activates, it is necessary to use the optimization model described in the above relation, because this model optimizes energy consumption in the network. In the prosed improved dragonfly optimization algorithm, each dragonfly signifies a solution with a conforming cost value, which results in the cost function. The cost function (objective function) is utilized to find the quality of the solution. After applying the algorithm, a new population with better members is produced. Additionally, the process stops when it reaches the stopping condition.

The better coding of individuals can increase the algorithm's efficiency. To optimize the energy-aware coverage, it is necessary to code the scheduling of the nodes. This research employs optimal node scheduling to activate and deactivate wireless sensor nodes at a definite time in such a way as to raise the network's lifetime. For programming, the solutions are displayed as binary strings. For instance, 0 indicates that the sensor node is inactive (eliminated), and 1 indicates that the node is active. The binary illustration of individuals in the improved dragonfly optimization algorithm for energy-aware coverage optimization is shown in Figure 3.

	$l_{i,1}$	$l_{i,2}$	$l_{i,3}$	$l_{i,j}$	....	$l_{i,N}$
1	1	**0**	0	1	.....	1
2	0	1	0	0	....	0
i	1	1	1	0	....	1
Γ	0	1	1	1	....	0

Figure 3. Binary definition of individuals in the improved dragonfly optimization algorithm for energy-aware coverage optimization.

In this figure, Γ represents the total quantity of members (candidates) and $\ell_{i,j}$ signifies the state of the node c_i in individual k. It should be noted that the length of each individual is equal to N, which means the number of sensor nodes. Since the size of the population causes the production of diverse communities, it is necessary to choose an appropriate size of the population. For the sake of simplification, the population size is determined before

running the algorithm; therefore, the size of the population is constant in each iteration of the algorithm. In Figure 4, 16 nodes are used to sense the environment, and there are several additional nodes (15) (red circles).

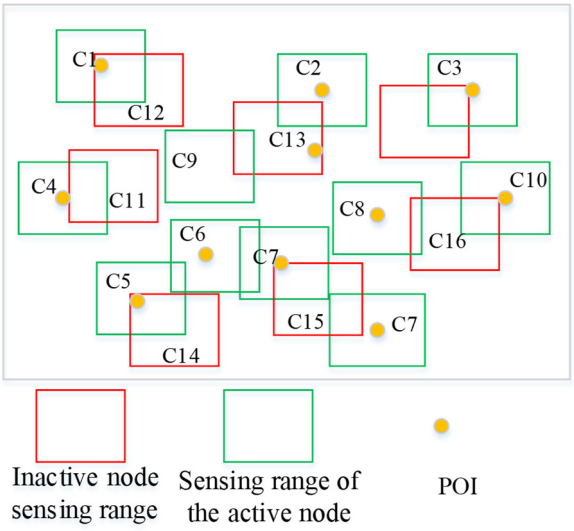

Figure 4. Wireless sensor network coverage with 16 active nodes.

To conserve energy and obtain the optimal coverage ratio, there is an optimal scheduling of nodes in the algorithm, which disables the redundant nodes. From the point of view of encoding particles, string 1111111111000000 represents the example in Figure 4.

The fitness function has been utilized to measure the quality of the candidates. The purpose of the algorithm is to obtain the optimal coverage ratio using the least amount of sensor nodes. For nodes, a coverage vector has been suggested that represents the POIs' coverage.

As stated by the coverage model defined previously, the coverage vector of a specific node c_i is demarcated by $\pi = [LO_{i,1}, LO_{i,2}, \ldots, LO_{i,M}]$. A binary model is used to know the coverage of a POI by a wireless sensor. Boolean operators are applied to vectors. A composite covering vector of c_i and c_j can be obtained by the Boolean union between π_i and π_j:

$$\overline{w}(c_i, c_j) = \pi_i \vee \pi_j = \left[LO_{i,1} \vee LO_{j,1}, LO_{i,2} \vee LO_{j,2}, \ldots, LO_{i,M} \vee LO_{j,M}\right] \tag{22}$$

where, $\overline{w}$ is a combined coverage vector that shows the coverage or non-coverage of all POIs by c_i and c_j.

The combined coverage vector for a particular individual is calculated by the following equation:

$$\overline{w}(k) = (\ell_{k,1} \cdot \pi_1) \vee (\ell_{k,2} \cdot \pi_2) \vee \ldots \vee (\ell_{k,N} \cdot \pi_N) \tag{23}$$

The process of checking the coverage has been simplified to binary operators. The coverage ratio for every one, k, is:

$$\varepsilon^k = \frac{\|\omega(k)\|^2}{M} \tag{24}$$

where, $\|\omega(k)\|^2$ indicates the active nodes quantity. The nodes utility ratio, U_t^k, for individual k is evaluated as follows:

$$U_t^k = \frac{\sum_{g=1}^{M} \ell_{k,g}}{N} \tag{25}$$

The nominator of the relation signifies the nodes quantity selected for activation. The quality of individual k is defined by f_c^k:

$$f_c^k = \alpha_1 \times \left(\varepsilon^k\right)^{\lambda_1} - \alpha_2 \times \left(U_t^k\right)^{\lambda_2} \tag{26}$$

Substituting ε^k and U_t^k from the above equations, we have:

$$f_c^k = \alpha_1 \times \left(\frac{\|\omega(k)\|^2}{M}\right)^{\lambda_1} - \alpha_2 \times \left(\frac{\sum_{g=1}^{M} \ell_{k,g}}{N}\right)^{\lambda_2} \tag{27}$$

where, α_1 and α_2 are the weighting constants, and λ_1 and λ_2 represent the power factors.

5. Simulation Results

In the implementation and evaluation, we used an Intel® Core ™ i5-4460 CPU @3.20 GHz processor, 12 gigabytes of memory, and a Windows 10 (64-bit) system. The efficiency of the proposed method was evaluated using MATLAB software. The algorithm of the proposed method was implemented in this software, and simulations were carried out at different stages. MATLAB optimization functions were used in the process. The simulation work was repeated up to 2500 steps until the results reach a stable point.

In this study, we compared the proposed method with three methods from the literature, including the LEACH [35], RED-LEACH [36], and gravitational search algorithm (GSA)-based methods [37]. The LEACH method is one of the first clustering methods in sensor networks. In the proposed method, four parameters are used to evaluate the efficiency:

- The quantity of nodes in the exposure zone: The coverage area includes the area where the nodes are located.
- Network lifetime: The network lifetime includes the time from the beginning of the first node to the time when the first node dies.
- Network capacity: The ratio of the number of packets received by the well to the quantity of packets sent by the nodes.
- Residual energy: The remaining energy in the battery of the sensor nodes is used to analyze the energy consumption in each method.

In this scenario (Figure 5), the sink is placed in the center of a WSN.

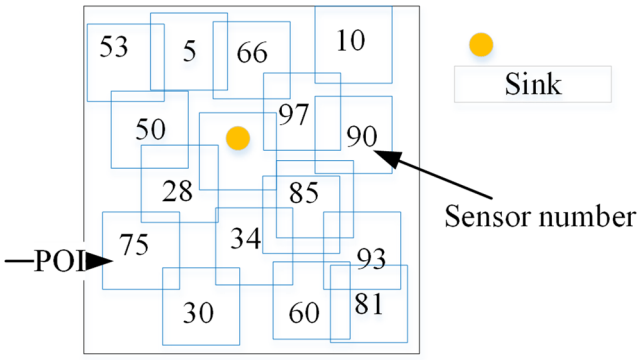

Figure 5. Coverage scenario in a WSN using the suggested algorithm.

Nodes are clustered, and each cluster head sends data directly to the sink without the use of an auxiliary node. The objectification parameters of this scenario are explained in the following information: The size of the grid in this study is set to 100 × 100 m^2. The main station location is (60, 120). The sensor node quantity is 110. The size of the data package is set at 500 bytes, E_{el} is set 50, the $\mathcal{E}_{fs}$ is set 10, the value of the $\mathcal{E}_{amp}$ is considered 0.0010, and the initial energy is set at 0.5 joules.

5.1. The Number of Nodes in the Coverage Area

The sensor nodes are usually randomly distributed in a network, and the area covered by them is shown with a circle. In Figure 5, the proposed improved dragonfly optimization algorithm is applied to this network. As seen in Figure 5, the coverage area and the number of active nodes has been optimized. Decreasing the number of active sensor nodes leads to a decrease in energy consumption and, as a result, to an increase in the lifetime of the network.

5.2. Network Lifetime

The number of live nodes is shown in Figure 6. These numbers are for the proposed method, the LEACH method [23], the RED-LEACH method [27], and the gravitational search algorithm (GSA)-based method [37]. In this diagram, the initial energy of the sensor nodes is assumed to be half a joule. Figure 6 shows the network lifetime based on the analyzed methods, and Figure 7 shows the number of off nodes.

It can be perceived from Figure 7 that, by increasing the number of steps, the energy of the live nodes will eventually run out and turn off. In the LEACH method, after 350 steps, the RED-LEACH method, after 750 steps, and the GSA-based method, after 915 steps, the nodes start shutting down, which occurs after 1227 steps for the proposed method. This means that the nodes are turned off later. Since the first node's shutdown time is considered in defining the lifetime of the network, this figure indicates that the suggested method offers a longer lifetime. The reason for this improvement is that the proposed method distributes the consumed energy in a balanced way among the network nodes. On the other hand, by increasing the number of steps, the percentage of live nodes in the proposed method works better than that of the other methods. This method means that after almost 2155 steps all of its nodes are turned off, and, until then, the probability of establishing a route for sending packets is higher than that of the other methods.

As can be seen in Figure 6, all of the network nodes in the LEACH method lost their energy after approximately 1418 steps, after 1859 steps for the RED-LEACH method, and after 1971 steps for the GSA-based method, while in the proposed method this event occurs after 2155 steps. This makes the nodes active in the network for a longer period of time and raises the network's lifespan.

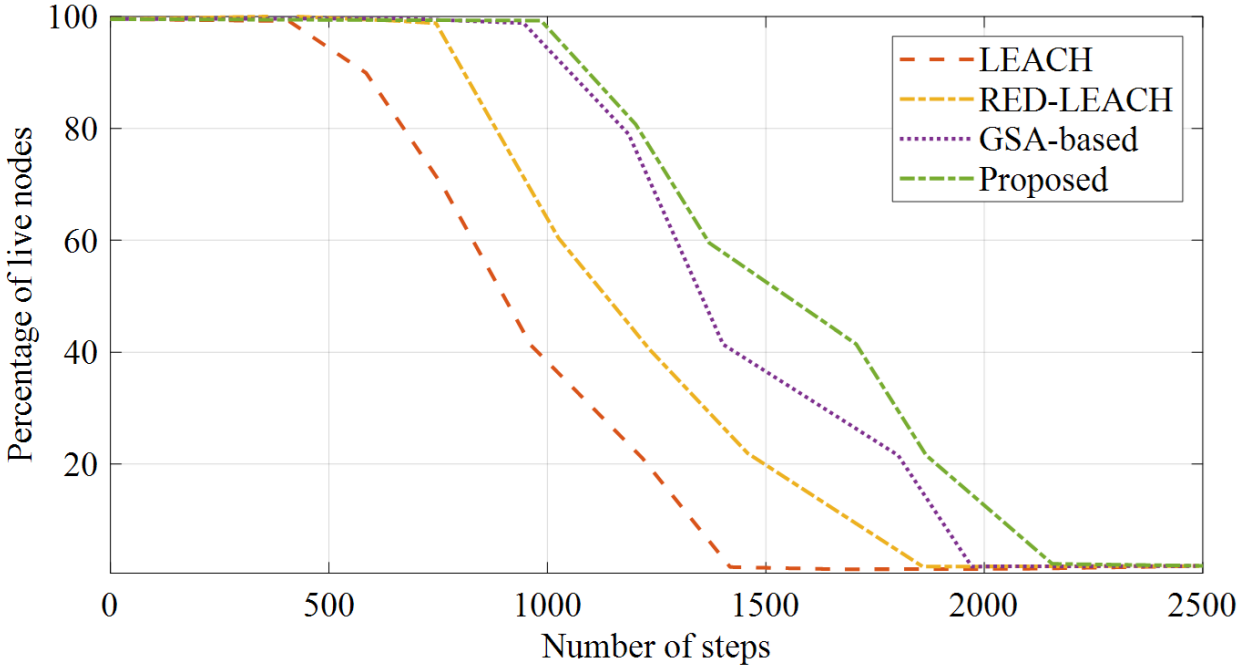

Figure 6. Network lifetime based on the analyzed methods.

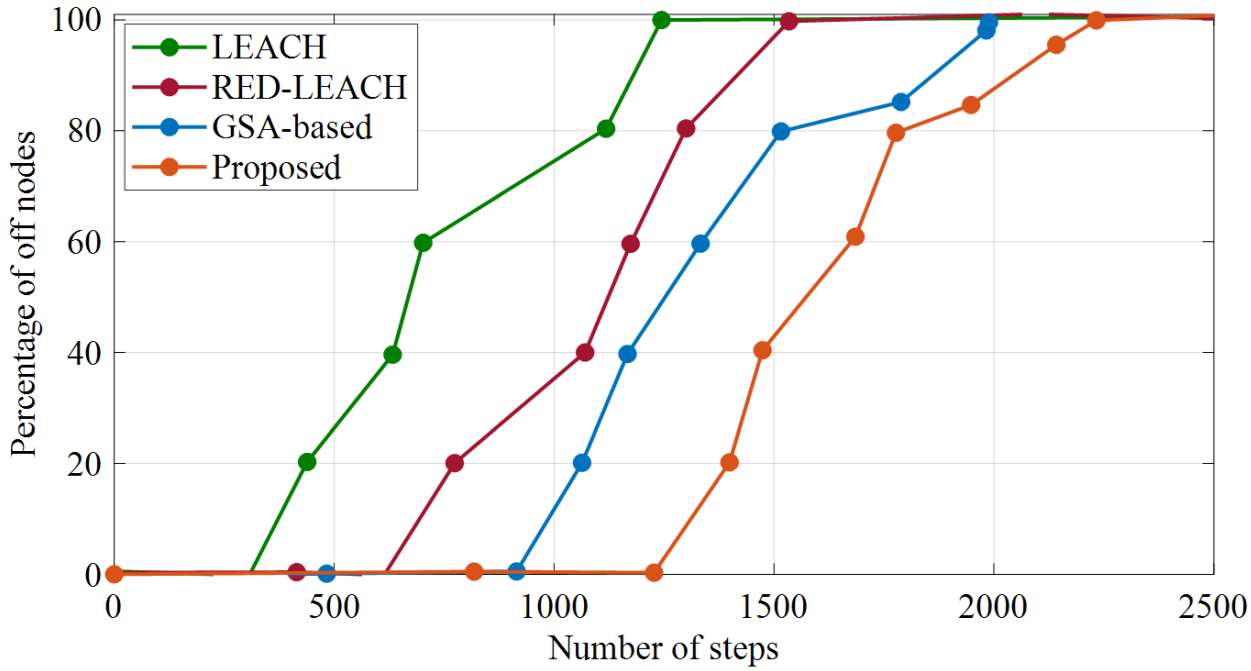

Figure 7. The number of off nodes.

5.3. Throughput (Network Power)

The throughput can be calculated based on two criteria, including the average received packets in the network and the average sent packets to the sink. The averages of the sent and received packets to the sink node were obtained during repeated simulations for the assessed methods. The simulation outcomes showed that the proposed method increased the power of the network. These results are shown in Figures 8 and 9.

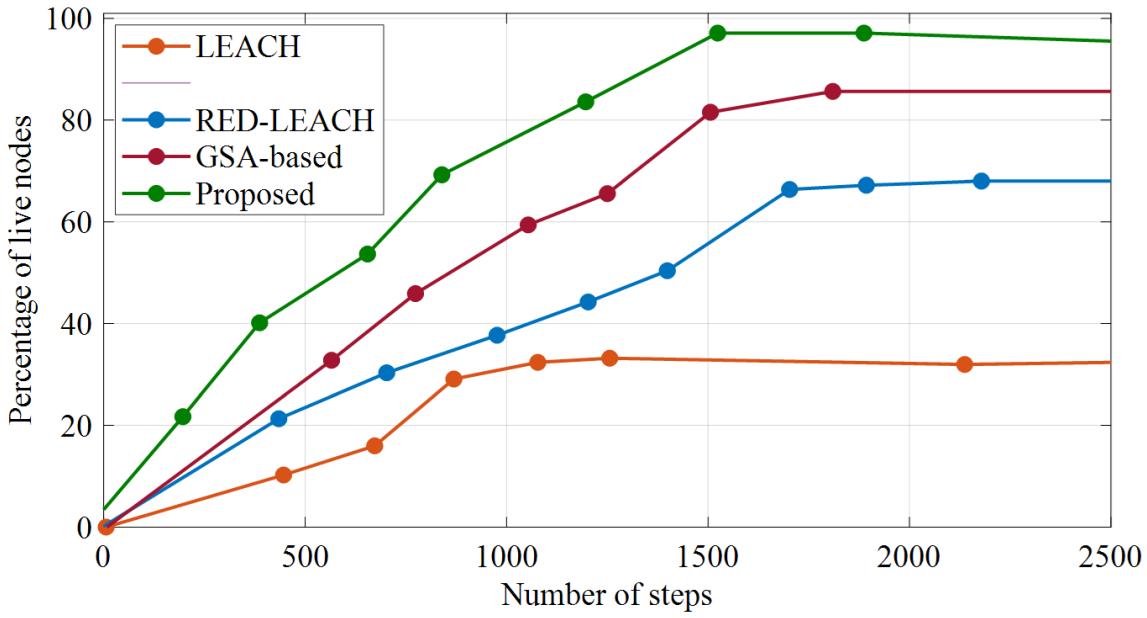

Figure 8. Throughput-sent packets to the sink.

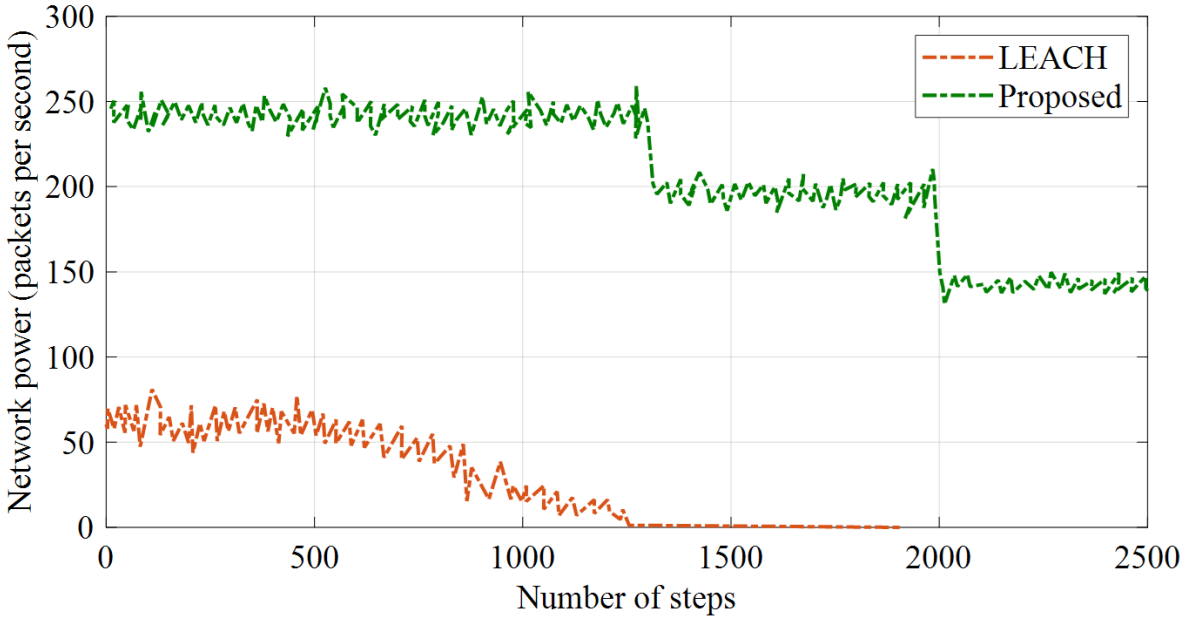

Figure 9. Throughput-received packets from the network.

5.4. Residual Energy

Figure 10 shows the average residual energy in each stage. Here, it is assumed that each node has an initial energy of half a joule. In this case, the network total energy with 110 sensor nodes is equal to 50 joules.

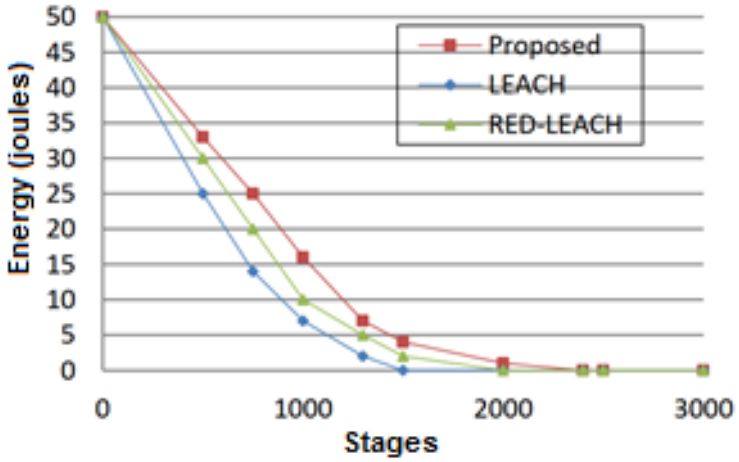

Figure 10. Average residual energy in each stage.

It can be concluded from Figure 10 that the remaining energy in the network decreases with an increase in the number of steps. The slope of this reduction is lower in the proposed technique. Accordingly, this method will have a longer network life. Additionally, the total energy of the network in the LEACH method reaches zero after 1500 steps, after 2000 steps in the RED_LEACH method, after 2400 steps in the PID_LEACH method, and after 2000 steps in the GSA-based method. This means that the probability of sending packets in the suggested technique will be higher than that of the other assessed methods with an increase in the number of data transfer rates.

As is observed, the proposed method has several advantages for the detection of IoT outlier sensors; however, it has also some shortcomings. Some of these shortcomings have been explained previously:

- The issue is that there is not a specific way to solve it.
- There is a precise solution to the issue and carrying it out incurs computational costs.
- There is no determinate metaheuristic method that can solve any kind of similar problems.

6. Conclusions

The Internet of things is rarely discussed without a conversation about new ecosystem data and information. The intelligence and value of an Internet of things (IoT) system are using what may be learned from data, and sensors are the main sources of these data. The subject of collecting and sending information by objects specifically refers to sensors. Sensors can be used for sensing temperature, proximity, pressure, movement, position, humidity, light, air quality, or anything else. These sensors, coupled with Internet connectivity, enable them to automatically collect data from the environment, enabling stakeholders to make smarter decisions. Sensor networks are used in various fields, and there are primarily problems with sensor costs and battery replacements. Therefore, using the minimum number of sensors with full coverage and elimination is an incredibly important subject. In this paper, a new approach was attempted to detect outlier data and eliminate them in the Internet of things. Outlier sensors are one of the challenges of this type of network. In this paper, an improved version of the dragonfly optimization algorithm was used for optimal area coverage and energy consumption reduction. This paper used four parameters to evaluate its efficiency: the minimum quantity of nodes in the coverage area, the lifetime of the network, including the time interval from the start of the first node to the shutdown time of the first node, and the network power. The results of the suggested approach were compared with some published techniques. Simulations showed that the suggested technique achieved better results than the other investigated methods in terms of the provided performance parameters. The goal of future research is to construct IoT sensors that are appropriate for outlier detection modules (ODMs), analyze the sensors' produced spatial–temporal data in real time, and create a testable IoT prototype with an integrated ODM. The investigation of potential computational constraints for the ODM will also be a key component of the project. Additionally, the effectiveness of the IoT will be examined, as will its capacity to correctly identify any malfunctioning sensors as well as the type, source, and impact of errors on these sensors.

Author Contributions: Conceptualization, M.N.M. and S.K.; methodology, M.N.M.; software, H.T.R.; validation, M.N.M., S.K. and H.T.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Boursianis, A.D.; Papadopoulou, M.S.; Diamantoulakis, P.; Liopa-Tsakalidi, A.; Barouchas, P.; Salahas, G.; Karagiannidis, G.; Wan, S.; Goudos, S.K. Internet of Things (IoT) and Agricultural Unmanned Aerial Vehicles (UAVs) in smart farming: A comprehensive review. *Internet Things* **2022**, *18*, 100187. [CrossRef]
2. Al-Khafaji, H.M.R. Improving Quality Indicators of the Cloud-Based IoT Networks Using an Improved Form of Seagull Optimization Algorithm. *Future Internet* **2022**, *14*, 281. [CrossRef]
3. Alferaidi, A.; Yadav, K.; Alharbi, Y.; Razmjooy, N.; Viriyasitavat, W.; Gulati, K.; Kautish, S.; Dhiman, G. Distributed Deep CNN-LSTM Model for Intrusion Detection Method in IoT-Based Vehicles. *Math. Probl. Eng.* **2022**, *2022*, 3424819. [CrossRef]
4. Quy, V.K.; Hau, N.V.; Anh, D.V.; Quy, N.M.; Ban, N.T.; Lanza, S.; Randazzo, G.; Muzirafuti, A. IoT-Enabled Smart Agriculture: Architecture, Applications, and Challenges. *Appl. Sci.* **2022**, *12*, 3396. [CrossRef]
5. Koohang, A.; Sargent, C.S.; Nord, J.H.; Paliszkiewicz, J. Internet of Things (IoT): From awareness to continued use. *Int. J. Inf. Manag.* **2021**, *62*, 102442. [CrossRef]

6. Bahmanyar, D.; Razmjooy, N.; Mirjalili, S. Multi-objective scheduling of IoT-enabled smart homes for energy management based on Arithmetic Optimization Algorithm: A Node-RED and NodeMCU module-based technique. *Knowl. Based Syst.* **2022**, *247*, 108762. [CrossRef]
7. Samara, M.; Bennis, I.; Abouaissa, A.; Lorenz, P. A Survey of Outlier Detection Techniques in IoT: Review and Classification. *J. Sens. Actuator Netw.* **2022**, *11*, 4. [CrossRef]
8. Gupta, D. Prediction of Sensor Faults and Outliers in IoT Devices. In Proceedings of the 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), IEEE, Noida, India, 3–4 September 2021; pp. 1–5.
9. Huang, X. Implementation and Verification of Outlier Data Classyfication for the Local Sensor of Atmospheric Pressure with the Use of IoT Technology. Bachelor's Thesis, Instytut Mikroelektroniki i Optoelektroniki, Warsaw, Poland, 2022.
10. Brahmam, M.V.; Gopikrishnan, S.; Kumar, K.R.S.; Bhavani, M.S. Pearson Correlation Based Outlier Detection in Spatial-Temporal Data of IoT Networks. In *Innovative Data Communication Technologies and Application*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 96, pp. 1019–1028.
11. Hajikarimi, A.; Bahaghighat, M. Optimum outlier detection in Internet of things industries using autoencoder. In *Frontiers in Nature-Inspired Industrial Optimization*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 77–92.
12. Foughali, A.; Kitouni, I.; Benmerzoug, D. ODMR-IoT: Outliers Detection Based Multipath Routing Protocol for Internet of Things (IoT). *Ingénierie Des Syst. D'inf.* **2022**, *27*, 377–385. [CrossRef]
13. Bashir, A.; Awawdeh, M.; Faisal, T.; Queen, M.F. Matlab-based Graphical User Interface for IoT Sensor Measurements Subject to Outlier. In Proceedings of the 2022 Advances in Science and Engineering Technology International Conferences (ASET), IEEE, Dubai, United Arab Emirates, 21–24 February 2022; pp. 1–6.
14. Wei, Z.; Wang, F. Detecting Anomaly Data for IoT Sensor Networks. *Sci. Program.* **2022**, *2022*, 4671381. [CrossRef]
15. Boukela, L.; Zhang, G.; Yacoub, M.; Bouzefrane, S.; Ahmadi, S.B.B.; Jelodar, H. A modified LOF-based approach for outlier characterization in IoT. *Ann. Telecommun.* **2020**, *76*, 145–153. [CrossRef]
16. Safaei, M.; Asadi, S.; Driss, M.; Boulila, W.; Alsaeedi, A.; Chizari, H.; Abdullah, R.; Safaei, M. A systematic literature review on outlier detection in wireless sensor networks. *Symmetry* **2020**, *12*, 328. [CrossRef]
17. Deng, X.; Jiang, P.; Peng, X.; Mi, C. An Intelligent Outlier Detection Method With One Class Support Tucker Machine and Genetic Algorithm Toward Big Sensor Data in Internet of Things. *IEEE Trans. Ind. Electron.* **2018**, *66*, 4672–4683. [CrossRef]
18. Titouna, C.; Naït-Abdesselam, F.; Khokhar, A. DODS: A Distributed Outlier Detection Scheme for Wireless Sensor Networks. *Comput. Netw.* **2019**, *161*, 93–101. [CrossRef]
19. Ferrer-Cid, P.; Barcelo-Ordinas, J.M.; Garcia-Vidal, J. Volterra Graph-Based Outlier Detection for Air Pollution Sensor Networks. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 2759–2771. [CrossRef]
20. Dwivedi, R.K.; Pandey, S.; Kumar, R. A Study on Machine Learning Approaches for Outlier Detection in Wireless Sensor network. In Proceedings of the 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), IEEE, Noida, India, 11–12 January 2018; pp. 189–192.
21. Gil, P.; Martins, H.; Januário, F. Outliers detection methods in wireless sensor networks. *Artif. Intell. Rev.* **2019**, *52*, 2411–2436. [CrossRef]
22. Alshinwan, M.; Abualigah, L.; Shehab, M.; Elaziz, M.A.; Khasawneh, A.M.; Alabool, H.; Hamad, H.A. Dragonfly algorithm: A comprehensive survey of its results, variants, and applications. *Multimed. Tools Appl.* **2021**, *80*, 14979–15016. [CrossRef]
23. Lodhi, E.; Wang, F.-Y.; Xiong, G.; Mallah, G.A.; Javed, M.Y.; Tamir, T.S.; Gao, D.W. A Dragonfly Optimization Algorithm for Extracting Maximum Power of Grid-Interfaced PV Systems. *Sustainability* **2021**, *13*, 10778. [CrossRef]
24. Wang, L.; Shi, R.; Dong, J. A Hybridization of Dragonfly Algorithm Optimization and Angle Modulation Mechanism for 0-1 Knapsack Problems. *Entropy* **2021**, *23*, 598. [CrossRef]
25. Urooj, S.; Alrowais, F.; Kuppusamy, R.; Teekaraman, Y.; Manoharan, H. New Gen Controlling Variable Using Dragonfly Algorithm in PV Panel. *Energies* **2021**, *14*, 790. [CrossRef]
26. Acı, Ç.İ.; Gülcan, H. A modified dragonfly optimization algorithm for single-and multiobjective problems using Brownian motion. *Comput. Intell. Neurosci.* **2019**, *2019*, 6871298. [CrossRef]
27. Meraihi, Y.; Ramdane-Cherif, A.; Acheli, D.; Mahseur, M. Dragonfly algorithm: A comprehensive review and applications. *Neural Comput. Appl.* **2020**, *32*, 16625–16646. [CrossRef]
28. Zhang, G.; Xiao, C.; Razmjooy, N. Optimal parameter extraction of PEM fuel cells by meta-heuristics. *Int. J. Ambient Energy* **2020**, *43*, 2510–2519. [CrossRef]
29. Razmjooy, N.; Razmjooy, S.; Vahedi, Z.; Estrela, V.V.; de Oliveira, G.G. A New Design for Robust Control of Power System Stabilizer Based on Moth Search Algorithm. In *Metaheuristics and Optimization in Computer and Electrical Engineering*; Springer: Berlin/Heidelberg, Germany, 2021; Volume 696, pp. 187–202.
30. Yin, Z.; Razmjooy, N. PEMFC identification using deep learning developed by improved deer hunting optimization algorithm. *Int. J. Power Energy Syst.* **2020**, *40*, 189–203. [CrossRef]
31. Ramezani, M.; Bahmanyar, D.; Razmjooy, N. A New Improved Model of Marine Predator Algorithm for Optimization Problems. *Arab. J. Sci. Eng.* **2021**, *46*, 8803–8826. [CrossRef]
32. Yang, D.; Li, G.; Cheng, G. On the efficiency of chaos optimization algorithms for global optimization. *Chaos Solitons Fractals* **2007**, *34*, 1366–1375. [CrossRef]

33. Rim, C.; Piao, S.; Li, G.; Pak, U. A niching chaos optimization algorithm for multimodal optimization. *Soft Comput.* **2016**, *22*, 621–633. [CrossRef]
34. Razmjooy, N.; Estrela, V.V.; Loschi, H.J.; Fanfan, W. A comprehensive survey of new meta-heuristic algorithms. In *Recent Advances in Hybrid Metaheuristics for Data Clustering*; Wiley Publishing: Hoboken, NJ, USA, 2019; pp. 1–25.
35. Rajput, M.; Sharma, S.K.; Khatri, P. Performance analysis of leach based approaches for large area coverage in wireless sensor network. In Proceedings of the 2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC), IEEE, Indore, India, 17–19 August 2017; pp. 1–5.
36. Chit, T.A.; Zar, K.T. Lifetime improvement of wireless sensor network using residual energy and distance parameters on LEACH protocol. In Proceedings of the 2018 18th International Symposium on Communications and Information Technologies (ISCIT), IEEE, Bangkok, Thailand, 26–29 September 2018; pp. 1–5.
37. Rezaee, A.A.; Zahedi, M.H.; Dehghan, Z. Coverage optimization in wireless sensor networks using gravitational search algorithm. *J. Soft Comput. Inf. Technol.* **2019**, *8*, 20–31.