REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

CLUSTER BASED ROUTING BY USING MFO META-HEURISTIC ALGORITHM

Ruwaida MAMOORI

MASTER OF SCIENCE THESIS

Department of Computer Engineering

Computer Engineering Program

Supervisor

Prof. Dr. Hasan Hüseyin BALIK

April, 2023

REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

CLUSTER BASED ROUTING BY USING MFO META-HEURISTIC ALGORITHM

A thesis submitted Ruwaida MAMOORI in partial fulfillment of the requirements for the degree of MASTER is approved by the committee on 17 .04 .2023 in Department of Computer Engineering, Computer Engineering Program.

Prof. Dr. Hasan Hüseyin BALIK
İstanbul Aydın University
Supervisor

Approved By the Examining Committee

Prof. Dr. Hasan Hüseyin BALIK, Supervisor İstanbul Aydın University	________________________
Prof. Dr. Nizamettin AYDIN , Member Yildiz Technical University	________________________
Doç. Dr. Can ETÜPOĞLU , Member National Defense University	________________________

My supervisor, Prof. Dr. Hasan Hüseyin BALIK, stated that in the study titled, Cluster-based routing by using MFO Meta-Heuristic Algorithm prepared by me under his responsibility, I've included the material I've gathered from various sources in the main text as well as the references section, that I have not made any distortions and/or forgeries regarding the research data and results. I declare that I act in accordance with scientific research and ethical principles. I accept all legal consequences if my statement is proven to the contrary.

Ruwaida MAMOORI

Signature

Dedicated to my family

and my best friends

ACKNOWLEDGEMENTS

First and foremost, we give gratitude and praise to God, the Almighty, for the abundance of benefits He has bestowed upon us. I would like to convey my heartfelt appreciation to my research supervisor, Prof. Dr. Hasan Hüseyin BALIK, computer engineering department, Yildiz Technical University, for his guidance and support during this process. I am eternally thankful to my mother for her love, prayers, care, and sacrifices made on my behalf in order to educate and prepare me for my future. It is with great gratitude that I express my gratitude to my sisters, brothers, and friends for their love, understanding, prayers, and ongoing support while I try to finish my research project.

Ruwaida MAMOORI

TABLE OF CONTENTS

LIST OF SYMBOLS

i	Candidate answer
g_i	Cluster head g in the answer of candidate i
D(gi)	Data transfer delay collected from gate gi to sink
d("CH,S")	Euclidean distance between the cluster head node and the sink
d("Mem,CH)"	Euclidean distance between the cluster member and the cluster head
D("i,j")	Euclidean distance between ith moths and jth flames
Di	Euclidean (ith) moth from (jth) flame
gfarthest	Farthest gateway from the sink
T	Function examines the end criterion of the algorithm
P	Function for relocating the moths throughout the search space
I	Function selects a set of initial candidate answers
j	Gateway
gj	Gatewayt is cluster head
$E_{\text{RX-elec}}(L)$	Initial energy required to receive L bits of the message
$E_{\text{TX-elec}}(L)$	Initial energy required to send L bits of the message
L (i)	Lifetime of the gate g_i
LBi	Lower bounds on the location of the moth i
LB	Lower borders of the moth location
R_{gmax}	Maximum range of information transmission in gateways
R_{smax}	Maximum range of information transmission in sensors
T	Maximum stage of execution of the MFO algorithm
dt	Message transmission delay
N_0	Number of cluster head members choose a gateway as the cluster head
Fn,d	Number of flames in which dimension for optimal response
HopCount(g_i)	Number of intermediate gateways from gateway gi to sink
N	Number of moths, flames
Mn,d	Number of moths in which dimension for optimal response
L	Number of recent execution steps
d	Number of search space dimensions for optimal response
d_p	Propagation delay
d_q	Queuing delay
PNextHops(g_i)	Set of gateways can be selected as the next node

$E_{Gateway}(g_i)$	The energy consumed in each turn by the gateway g_i
ε_{fs}	The energy required in free space channel transmission model
ε_{mp}	The energy required in multipath fading model
E_{RX}	The energy required to receive the message
E_{TX}	The energy required to send the message
u_x	The highest limits of the location on the x coordinate axes
u_y	The highest limits of the location on the y coordinate axes
l_x	The lowest limit of the location on the x coordinate axes
l_y	The lowest limit of the location on the y coordinate axes
$E_{residual}(g_i)$	The remaining energy
UBi	Upper bounds on the location of the moth i
UB	Upper borders of the moth location Farthest gateway from the sink
OF	Usefulness of flames
OM	Usefulness of the moths
$R(M_{is})$	Value of failure tolerance for gate g_j
OF_n	Value of the usefulness of n flame
Om_n	Value of the usefulness of the n moths

LIST OF ABBREVIATIONS

BA	Bat Algorithm
CH	Cluster head
ERX	The energy required to receive the message
ETX	The energy required to send the message
F	Matrix of flame set
FA	Firefly Algorithm
FBA	Flower pollination algorithm
GA	genetics algorithm
GAbEERP	Genetic Algorithm based Energy Efficient Routing Protocol
GLBCA	Greedy Load-Balanced Clustering Algorithm
GoGS (gj)	Gateways of Gateway Set
GoSS (s_i)	Gateways of Sensor Set
GREM	Greedy Maximum Residual Energy
GSA	Gravitational Search Algorithm
LEACH	Low Energy Adaptive Clustering Hierarchy
M	Matrix of Moths set
MaxHop	Maximum number of steps
MFO	Moth Flame Optimization
PSO	Particle Swarm Optimization
Rc	The transmission radius of gateway.
S	Number of sensors in the network
SMS	States of Matter Search
WSNs	Wireless Sensor Networks

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Cluster Based Routing by Using MFO Meta-Heuristic Algorithm

Ruwaida MAMOORI

Department of Computer Engineering

Master of Science Thesis

Supervisor: Prof. Dr. Hasan Hüseyin BALIK

This study presents a new routing protocol for WSNs called a cluster-based routing protocol inspired by the Meta-Heuristic Moth-Flame Algorithm, which is used in various applications such as predicting the weather, remote healthcare, and military information exchange. Protocol's primary objective is to increase the longevity of the network by focus on sensor networks' power consumption problem. The protocol is inspired by the Meta-Heuristic Moth-Flame Algorithm, which is an enhancement technique that is based on the attitude of moths towards a light source. The Moth-Flame Algorithm has been shown to be efficient in finding optimal solutions in WSNs. The proposed protocol uses unbalanced clustering techniques to prevent the formation of energy holes, which can lead to the early death of nodes and data transfer issues. Unbalanced clustering involves calculating the cluster size depending on how far away each cluster is from the sink. If a cluster is located near to the sink, it will be smaller, and if it is farther away, it will be bigger and this helps to prevent the formation of energy holes. The proposed protocol is compared to a Particle Swarm Algorithm, which is another commonly used optimization technique in WSNs. The Particle Swarm Algorithm is based on the behavior of a swarm of birds or fish. In the algorithm, each particle stands in for a potential answer and the swarm navigates the search area to locate the best answer. The results of the evaluation show that our method inspired by the Meta-Heuristic Moth-Flame Algorithm improves energy consumption and network longevity significantly when compared to the PSO

Algorithm and this suggests that our proposed protocol is an effective for this parameter in WSNs.

Keywords: WSN, Meta-Heuristic Algorithm's, Clustering, Routing Protocol's, Energy Consumption.

ÖZET

MFO Meta- Sezgisel Algoritma Kullanarak Küme Tabanlı Yönlendirme

Ruwaida MAMOORI

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Danışman: Prof. Dr. Hasan Hüseyin BALIK

Bu çalışma, WSN'ler için yeni bir yönlendirme protokolü sunmaktadır Meta- sezgisel Güve-Alev Algoritmasından ilham alan küme tabanlı bir yönlendirme protokolü olarak adlandırılır, hava tahmini gibi çeşitli uygulamalarda kullanılan, uzaktan sağlık hizmeti, ve askeri bilgi alışverişi. Protokolün birincil amacı, sensör ağlarının güç tüketimi sorununa odaklanarak ağın ömrünü artırmaktır. Protokol, güvelerin bir ışık kaynağına karşı tutumuna dayanan bir geliştirme tekniği olan Meta- sezgisel Güve-Alev Algoritmasından esinlenmiştir. Güve-Alev Algoritmasının WSN'lerde optimal çözümler bulmada etkili olduğu gösterilmiştir. Önerilen protokol, düğümlerin erken ölümüne ve veri aktarım sorunlarına yol açabilecek enerji boşluklarının oluşumunu önlemek için dengesiz kümeleme teknikleri kullanır. Dengesiz kümeleme, her bir kümenin havuzdan ne kadar uzakta olduğuna bağlı olarak küme boyutunun hesaplanmasını içerir. Lavaboya yakın bir küme bulunursa, daha küçük olacaktır ve eğer daha uzaksa, daha büyük olacaktır ve bu da enerji deliklerinin oluşumunu engellemeye yardımcı olur. Önerilen protokol, WSN'lerde yaygın olarak kullanılan başka bir optimizasyon tekniği olan Parçacık Sürü Algoritması ile karşılaştırılmıştır. Parçacık Sürü Algoritması, bir kuş veya balık sürüsünün davranışına dayanır. Algoritmada, her parçacık olası bir yanıt için duraklar ve sürü, en iyi yanıtı bulmak için arama alanını tarar. Değerlendirme sonuçları, Meta- sezgisel Güve-Alev Algoritmasından ilham alan yöntemimizin, PSO Algoritmasına kıyasla enerji tüketimini ve ağ ömrünü önemli ölçüde iyileştirdiğini göstermektedir ve bu, önerdiğimiz protokolün WSN'lerde bu parametre için etkili olduğunu göstermektedi.

Anahtar Kelimeler: WSN, Meta-Sezgisel Algoritma, Kümeleme, Yönlendirme Protokolleri, Enerji tüketimi.

1 INTRODUCTION

1.1 Literature Review

Here, we'll offer an introduction to algorithms and explain how they may be broadly categorized into two types: distributed and centralized. This section goal is to offer a high-level overview of the many kinds of algorithms and the features they share.

Distributed algorithms are built to run on a network of nodes, where each node functions independently and collaboratively with the other nodes and are built to be scalable and resilient so that they can operate in a network of any size and can tolerate node failures or changes in network architecture.

Centralized algorithms depend on a single node to handle all computing and network management, these algorithms are often easier to implement and more efficient to use, but they are less stable and scalable than distributed algorithms due to their dependence on a single processing node.

The MFO and PSO algorithms belong to the same family of meta-heuristic optimization algorithms known as distributed or decentralized control algorithms, these algorithms rely on a swarm intelligence technique to get optimal results, and they are meant to function independently of a central processing unit. Both PSO particles and MFO moths operate in parallel to solve the problem at hand without an overseer or leader and these algorithms are useful in big and complicated systems thanks to the decentralized approach, which may not be possible with a centralized method.

1.1.1 Distributed Control Algorithms

[1] provided an uneven and effective energy clustering mechanism for WSNs and they called their suggested mechanism (EEUC). In this mechanism, the issue of hot spots on wireless sensor networks has taken a few steps. Hot spots create energy hole in the wireless sensor networks multi-hop. Energy hole means premature death of sensors located near the sink and during its transmission radius and prevent data transfer to sink. To this end, in this mechanism, the heads of the clusters determine their clusters according

to measure of how far away the radius station is. The shorter the cluster's head than the Sink, the reduced the cluster radius.

In this algorithm, it is supposed that the nodes are evenly distributed in the network space. The sensors are fixed and the sink is outside the network. All sensors have the same initial energy. Energy consumption in this algorithm is obtained using equations (1.1) and (1.2) in algorithm. Each sensor obtains its competitive radius using equation (1.3).

$$E_{\mathrm{TX}}(\mathrm{b,d}) = \mathrm{E_{elec}} \times \mathrm{b} + \varepsilon_{\mathrm{amp}} \times b \times d^{\lambda} \tag{1.1}$$

$$E_{\mathrm{Rx}}(b) = \mathrm{E_{elec}} \times b \tag{1.2}$$

Equation (1.1) shows the energy required to transmit b data bits at distance d. In this regard, it indicates path loss in the network. Equation (1. 2) shows the amount of power needed to receive b bits of data.

$$s_i.R_{\mathrm{comp}} = (1 - (f \times (\frac{d_{\max} - d(s_i,\mathrm{sink})}{d_{\max} - d_{\min}})) \times R_{\mathrm{comp}}^{0} \tag{1.3}$$

In equation (1.3) using for each sensor to obtains its competitive radius, f is a fixed number in the open range (1,0). $d_{\max}$ and $d_{\min}$ are the maximum and minimum distance of the sensors from the sink, respectively. Indicates the transmission radius of the sensor, which is constant for all sensors. Figure (1.1) shows an example of unbalanced clustering.

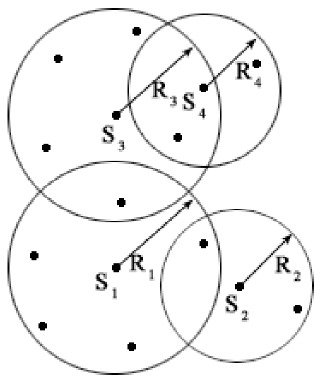


Figure 1. 1 Competitive radius of the clusters [1]

The clustering algorithm is performed independently in each node. First, every node picks its own random digit in the open interval (1,0) and sets it to ‘μ’. It then checks whether the value of ‘μ’ is less than the threshold ‘T’. If the estimation of ‘μ’ is lesser than the threshold ‘T’, it chooses itself as the head of the experimental cluster and the value becomes be, Tentative Head = True. Nodes that have selected themselves as experimental cluster heads generate a CompteteHeadMsg message and send their identification number, competitive radius, and residual energy to neighboring nodes. Each node first checks the sender's competitive radius as soon as it receives the CompteteHeadMsg message. If the transmitter node is within a competitive radius of the receiver node or vice versa, the receiving node of the CompteteHeadMsg message stores the sender node ID number plus its residual energy in the table of neighboring test clusters. It then checks after a period of time that if the last node is an experimental cluster head and its remaining power is greater than all the nodes in the neighboring experimental cluster head table, it selects itself as the final cluster head. If two or more experimental cluster heads have the most residual energy, the sensor with the lowest identification number is selected as the final head of cluster. Following the nodes that have been finalized as cluster heads, the FinalHeadMsg message is transmitted to all neighboring nodes. The FinalHeadMsg message includes the ID number and residual energy of the sender node. Each node that gets the FinalHeadMsg message first stores the identifier number and remaining power of the transmitter node in the final cluster head table of the neighbor. It then checks to see if it has previously selected itself as an experimental cluster head. Sends a QuitElectionMsg message to neighboring nodes if it chooses itself as an experimental cluster head. The message QuitElectionMsg contains the sender's sensor identification node. Each node removes the sender node from the neighboring test cluster table as soon as it receives the QuitElectionMsg message. Each final cluster head generates a CH_JOIN_MSG message and enters its ID number. It then distributes it to all neighboring nodes. If a node isn't chosen to lead a cluster, it will choose the node closest to the cluster's final leader and generates a JOIN_CLUSTER_MSG message with its ID number. It then sends it to the final selected cluster head.

[2] proposed a clustering algorithm with dual sink. This algorithm is distributive and increases network life and they call it dual sink (DS). In this algorithm there is a fixed sink and a mobile sink. In this method, a fixed sink spreads its location to all network sensors once at the beginning of the algorithm, but the mobile sink repeatedly informs its location only to neighboring sensors. Nodes that are not moving in the area of the sink also transfer informational messages to the fixed sink through the detected routing. This is shown in figure (1.2).

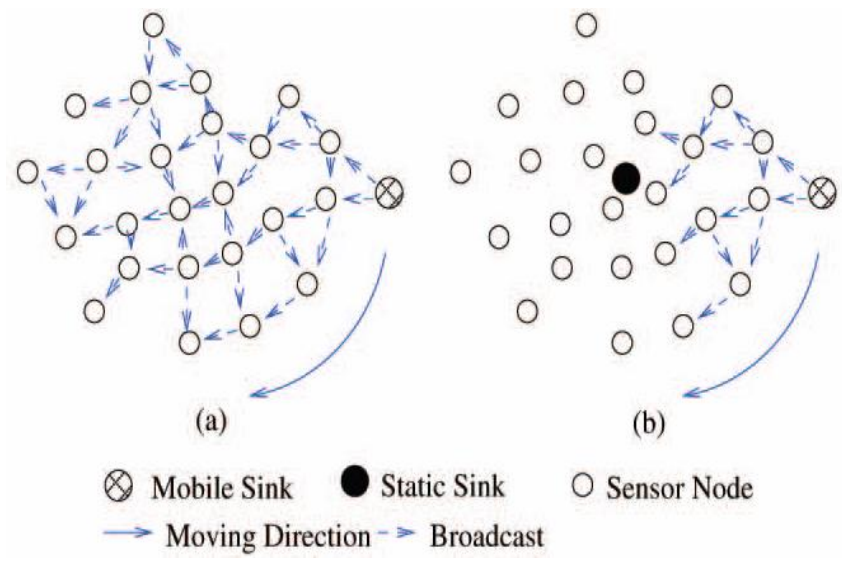


Figure 1. 2 (a): Mobile sink spreads its location to all sensor nodes (b): Mobile sink sends its location only to the sensors in its transmission space [2]

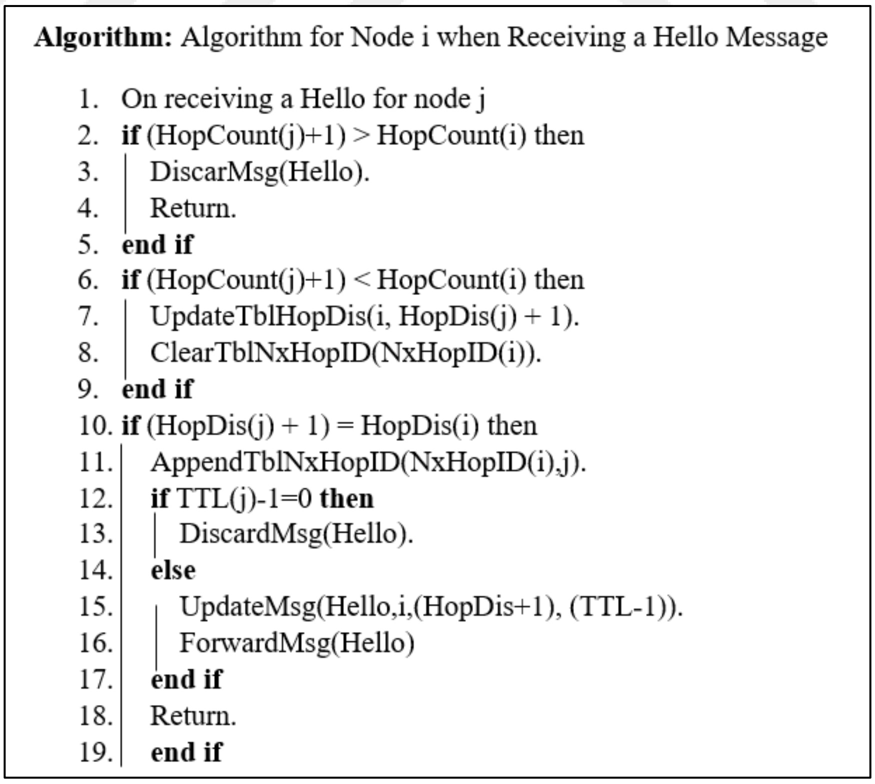
When both mobile and fixed sinks are within range of the sensor, the sensor node transfer its data closer to the sink. The operation on the fixed sink, mobile sink and sensors are described separately below.

- Fixed sink: At the beginning of the network setup, it will send out a welcoming message to every sensor in the system. These are the primary components of the "Hello" message:

1. Type of sink: mobile or fixed.
2. Identification number of the middle sensor of the message receiver.
3. Number of middle hops to fixed sink.
4. TTL: Used to restrict the transmission of all broadcast messages on the network. Initially, the TTL value is 10. Each intermediate node of the Hello message subtracts one TTL. When TTL is zero, the Hello message will be deleted and will no longer be broadcast to all neighboring nodes. TTL was set to 10 initially since it was considered to be suitable for the target network and the messages being sent. The TTL setting is meant to restrict how many times a broadcast message may be sent across the network. The message will only be sent to 10 nodes before being removed if the TTL is set to 10. As a result, the network's performance and reliability are preserved by a decrease in needless traffic. The number 10 was picked at random, and other networks may choose a different figure that better fits their needs and architecture.

- Mobile sink: Whenever the mobile sink pauses in a new location, it sends a greeting msg to the neighbors of one of its hops. It should be noted that this greeting message puts TTL = 1. It should be noted, however, that the length of time the sink stops at a location to receive data from neighboring sensors is large enough to receive all data sent from neighboring sensors. In other words, time required for a mobile sink to reach a new location is equal to the time period for retransmitting data from the sensors.
- Network sensors: Hello msg, data msg, energy request and energy response message are the messages that each sensor in this algorithm processes. The greeting message is received from a fixed or mobile sink. The data message is generated by the sensor and sent to the appropriate sink. Each sensor can also detect its neighboring sensors by sending EnQry and EnRpl messages.

In the proposed algorithm of [2], each sensor first checks whether it has received a greeting msg from the mobile sink. If it receives a greeting message from the mobile sink, it transfers the data directly to the mobile sink. Otherwise, it sends the data via the shortest path to the fixed sink detected by the routing algorithm. Figure (1.3) shows the pseudo-code of the proposed algorithm.



```
Algorithm: Algorithm for Node i when Receiving a Hello Message

On receiving a Hello for node j
if (HopCount(j)+1) > HopCount(i) then
    DiscarMsg(Hello).
    Return.
end if
if (HopCount(j)+1) < HopCount(i) then
    UpdateTblHopDis(i, HopDis(j) + 1).
    ClearTblNxHopID(NxHopID(i)).
end if
if (HopDis(j) + 1) = HopDis(i) then
    AppendTblNxHopID(NxHopID(i),j).
    if TTL(j)-1=0 then
        DiscardMsg(Hello).
    else
        UpdateMsg(Hello,i,(HopDis+1), (TTL-1)).
        ForwardMsg(Hello)
    end if
    Return.
end if
```

Figure 1. 3 Pseudo-code of the proposed algorithm [2]

Lines 2 to 5 in the pseudo-code indicate when the number of recent message hop from sensor j is greater than the minimum number of previous message hop. In this case, the i sensor ignores the greeting message. Lines 6 to 9 of the pseudo-code are for when the number of recent message hop is less than the j sensor, in which case the next sensor to transfer data to the sink is updated in the routing table and the previous sensor is removed. If the number of recent hops jumps from sensor j is similar to the minimum number of hops jumps in previous messages, the identification number of this sensor will be placed next to the sensors in the routing table to transfer data to the sink. In this case, if TTL-1 is equal to zero, the greeting message will be lost, otherwise the message will be re-broadcast to all neighboring nodes. The evaluation results show that the use of mobile sink along with fixed sink has increased scalability and network lifetime.

[3,4] have developed a cluster-based routing algorithm inspired by the Meta-Heuristic Particle Swarm Algorithm. This algorithm consists of two operational phases of routing and clustering. In both phases, a Particle Swarm Meta-Heuristic Algorithm is used for optimization. Both phases of the algorithm are performed in the sink. There are two major differences between routing and clustering phases:

- In the routing phase, each candidate answer is an array identical to the number of network gateways, while in the clustering phase, each candidate answer is an array equal to the number of network sensors.
- The purpose of the routing phase is to discover the finest gateway as the next hop to transfer data to the sink, while in the clustering phase it is to find the right gateway to transfer data. Obviously, the more energy the cluster head is and the closer it is to the sink, the higher the efficiency.

In routing algorithm, first r candidate answer is generated randomly. Each candidate answer is considered as a particle in the particle swarm meta-heuristic algorithm. The utility of each Pi particle is then obtained using the utility function of equation (1.4). In equation (1.4), MaxDistance and MaxNumber show the maximum distance and maximum number of steps from the sink in response to the candidate Qi, respectively.

$$\text{Fitness}_R(P_i) = W_1 \times \text{ MaxDistance}(P_i) + W2 \times \text{MaxNumber}(P_i) \qquad (1.4)$$

Initially, for each P_i particle, $Pbest_i$=P_i is considered, in which $Pbest_i$ is the optimal local answer for the **i**th particle. Also, the best particle in terms of usefulness is considered as Gbest. Then the number of iteration algorithms is considered equal to one. As long as the number of executions of the algorithm is less than or equal to the maximum of the implementation stage of the Max_iteration algorithm, the following is done.

- Particle velocities and positions are updated using equations (1.5) and (1.6), respectively.

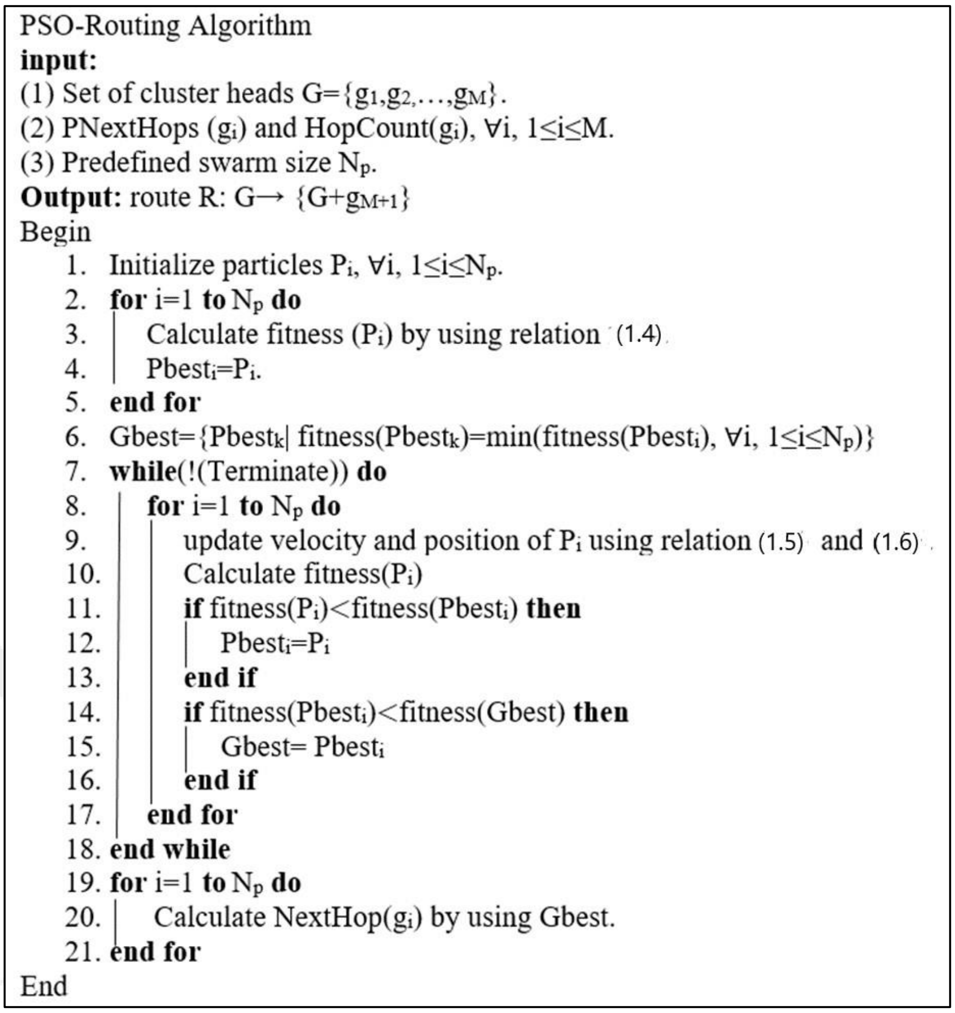
$$V_{i,d}(t)=\omega \times V_{i,d}(t\text{-}1)+c_1 \times r_1 \times (XPbest_{i,d} - X_{i,d}(t\text{-}1))+c_2 \times r_2 \times (XGbest_d - X_{i,d}(t\text{-}1)) \quad (1.5)$$

In equation (1.5), 'ω' is the inertial weight and is considered a constant number. c_1 and c_2 are two fixed numbers that are considered as acceleration factors to reach the answer of the final optimal candidate. r_1 and r_2 are two fixed numbers that are randomly selected from the open range (0,1).

$$X_{i,d}(t)=X_{i,d}(t\text{-}1)+V_{i,d}(t) \quad (1.6)$$

- The usefulness of new particles is obtained using equation (1.4).
- $Pbest_i$ and GBest will be updated.
- The number of algorithms implemented per unit increases.

Figure (1.4) shows routing algorithm based on PSO algorithm.



```
PSO-Routing Algorithm
input:
(1) Set of cluster heads G={g_1,g_2,...,g_M}.
(2) PNextHops (g_i) and HopCount(g_i), ∀i, 1≤i≤M.
(3) Predefined swarm size N_p.
Output: route R: G→ {G+g_{M+1}}
Begin
    1.  Initialize particles P_i, ∀i, 1≤i≤N_p.
    2.  for i=1 to N_p do
    3.      Calculate fitness (P_i) by using relation (1.4).
    4.      Pbest_i=P_i.
    5.  end for
    6.  Gbest={Pbest_k| fitness(Pbest_k)=min(fitness(Pbest_i), ∀i, 1≤i≤N_p)}
    7.  while(!(Terminate)) do
    8.      for i=1 to N_p do
    9.          update velocity and position of P_i using relation (1.5) and (1.6).
    10.         Calculate fitness(P_i)
    11.         if fitness(P_i)<fitness(Pbest_i) then
    12.             Pbest_i=P_i
    13.         end if
    14.         if fitness(Pbest_i)<fitness(Gbest) then
    15.             Gbest= Pbest_i
    16.         end if
    17.     end for
    18. end while
    19. for i=1 to N_p do
    20.     Calculate NextHop(g_i) by using Gbest.
    21. end for
End
```

Figure 1. 4 Routing algorithm based on PSO algorithm [3]

After the routing phase, the clustering phase is executed. In this phase, each sensor identifies an optimal cluster head for data transfer. The usefulness of each Pi particle in the clustering phase is obtained using equations (1.7), (1.8) and (1.9).

$$b_{\text{ij}} = \begin{cases} 1 & \text{if node sensor } \text{s}_i \text{ is allocated to cluster head } \text{g}_j\,, \\ & \forall\text{i,j: } 1 \le i \le \text{N},\ 1 \le j \le M \\ 0 & \text{otherwise} \end{cases} \tag{1.7}$$

$$\text{AvegDistance}=\frac{1}{N}\sum_{\text{i}=1}^{N}\sum_{\text{j}=1}^{M}\text{dis}(s_i, g_j) \times b_{\text{ij}} \tag{1.8}$$

$$\text{fitness}_C(z_i) = \frac{L}{\text{MaxDist}} \tag{1.9}$$

In equation (1.9), L is the minimum life and MaxDist is the maximum distance of the cluster heads from the sink. The pseudocode of the clustering phase is similar to the pseudocode of the routing phase, except that in the routing phase a gate is selected as the next intermediate node to transfer data to the sink, but in the clustering phase a gate is selected as the head of cluster.

[5] proposed a genetically based routing protocol to lengthen the lifespan of two-layer WSNs. They called this protocol Genetic Algorithm based Energy Efficient Routing Protocol (GAbEERP). The routing protocol dramatically increases the lifespan of the sensor in the network. When there is a small number of nodes in a network, this protocol provides the best possible solution. This routing protocol also offers a convenient solution when the number of network nodes is high. The proposed routing protocol consists of two operational phases, startup and steady state. The authors have envisioned a two-tier architecture in which higher-energy playback nodes are selected as the head of cluster. In this algorithm, it is assumed that the routing schedule is calculated by sink. Also, there is already an established knowledge of the data transfer rate between sensor nodes and of each node's membership in the clusters.

The suggested routing protocol seeks to optimize the lifespan of the sensor network by discovering the most efficient time to gather data. The lifetime of a network is calculated by the number of cycles. In other words, they used the N-of-N criterion (The amount of time before the first gateway fails) to determine the longevity of a network. In this criterion, if a single node in the network fails, the network will eventually fail. In the following, we will describe each operation of the GA algorithm to solve this problem in the best possible way. For the purposes of this routing protocol, the sink interprets the chromosomes as a sequence of integers representing the cluster heads of intermediary nodes. The number of middle cluster heads is proportional to the chromosome's length. Every chromosome in the initial population is equal to a valid path from the request node requesting a path to the sink. In the proposed algorithm, the initial routing is based on the position of the middle cluster head nodes. The sink uses the location of the intermediate nodes in routing a list of N = { N_1, N_2,…, N_i } in which N_i is the one-step neighbors of node i that have a path to the sink as the link $i \rightarrow j$, $\forall j \in N_i$ is in one of the existing paths from node i to sink. According to these findings, possible routing patterns for the initial population are created by a greedy method. In this method, each chromosome is obtained by randomly selecting a $j \in N_i$ node for each node requesting path i. Chromosome selection for Mutation and Crossover operations is done using the Roulette-Wheel. In this method, by increasing the suitability of the chromosome, the probability of its selection also increases. New offspring are generated from randomly selected parents using the uniform fragmentation function or k-point fragmentation.

The mutation function is used to improve the suitability of chromosomes. Unlike the standard genetic method, in which chromosomes are randomly selected for mutation operations, a node is selected here to perform a mutation on a chromosome that consumes more energy when sending or receiving information. Node i, which wastes a lot of energy, is called a critical node. The purpose of selecting node i for the mutation is to minimize network energy usage. After the production of each chromosome member, the value of the fitness function must be evaluated. The function's worth is determined by the expected lifetime of the network. The number of rounds is used to calculate this value. The value of the fit function for a chromosome is calculated using equation (1.10).

$$L_{\text{net}} = \frac{E_{\text{initial}}}{E_{\max}} \tag{1.10}$$

In equation (1.10), L_{net} represents the network lifespan based on the number of cycles on the chromosome. $E_{initial}$ also represents the initial energy of the node of cluster head. It is assumed that the $E_{initial}$ value is already known and its value is the same for all nodes. E_{max} is the maximum power consumed by the cluster head node.

1.1.2 Centralized Control Algorithms

To control mobile sink nodes, The authors first provide a (MILP) mixed-integer linear program [6]. The results of this model then obtain the paths leading to the sink that improve the network's reliability and extend its lifespan. Then they proposed an innovative algorithm to control the movement of sink nodes called (GREM) Greedy Maximum Residual Energy. The proposed algorithm moves the sink node from the current position to another position where the nodes have more energy. The authors also propose another algorithm called RM (Random Memory), in which the location of the sink node is constantly changing. First, they use a (MILP) that specifies the paths to the sink node and the position of the sink node. Also in this model, parameters such as the cost of mobile the sink node from one place to another are delayed in terms of energy consumption. The sink node's maximum allowed movement speed and total allowed movement duration are also constrained by this model. Although it is simple to include data routing optimization into the MILP model, the authors consider it independent of routing. Because a centralized routing solution is not suitable for sensor networks. In addition, considering routing as part of the model affects it in terms of parameters such as network lifespan. MILP models solve problems centrally. For example, to find paths with optimal lifetime for mobile sink nodes, one must have an overview of network topology, communication costs, and so on.

In the GREM protocol, the sink node greedily selects the nodes with the max remaining energy at the surrounding D_{max} distance. The main idea is that at any time, they have the potential to reduce overall network downtime and improve energy efficiency. After a time in a location, a sink node evaluates whether it moves to a nearby location or stays in place. Two locations are neighbors if the distance between them is greater than or equal to D_{max}. In order to start deciding whether to stay at the current position or move to another position, the sink node collects data about the amount of energy left by the nodes at nearby locations and compares it with the energy of the nodes inside the current location. If the energy of a neighboring place is higher than the current place, then it is transferred to the neighboring place. Otherwise, the sink node will remain in place. The main point in implementing the GREM algorithm is how the sink node communicates with neighboring locations to know the remaining energy of the nodes. This communication occurs in two stages. In the first stage, the sink node determines a sentinel sensor node for each adjacent location. The sentinel node is responsible for collecting information about the amount of power remaining in that location. The sentinel node then sends this information to the sink if requested. The first stage is implemented in such a way that the sink node uses the flood method to inform all nodes of their current location. For this exploratory protocol, it is assumed that the node adjacent to a location transfer is aware of the sink node. The all-broadcast message contains the current location of the sink node. When it receives a flood package, it will know if a node in the future is adjacent to the sink location. To illustrate this point, it sends a small package to the sink and introduces itself as a sentinel. The sink node will check if this packet will be guarded if it receives this packet.

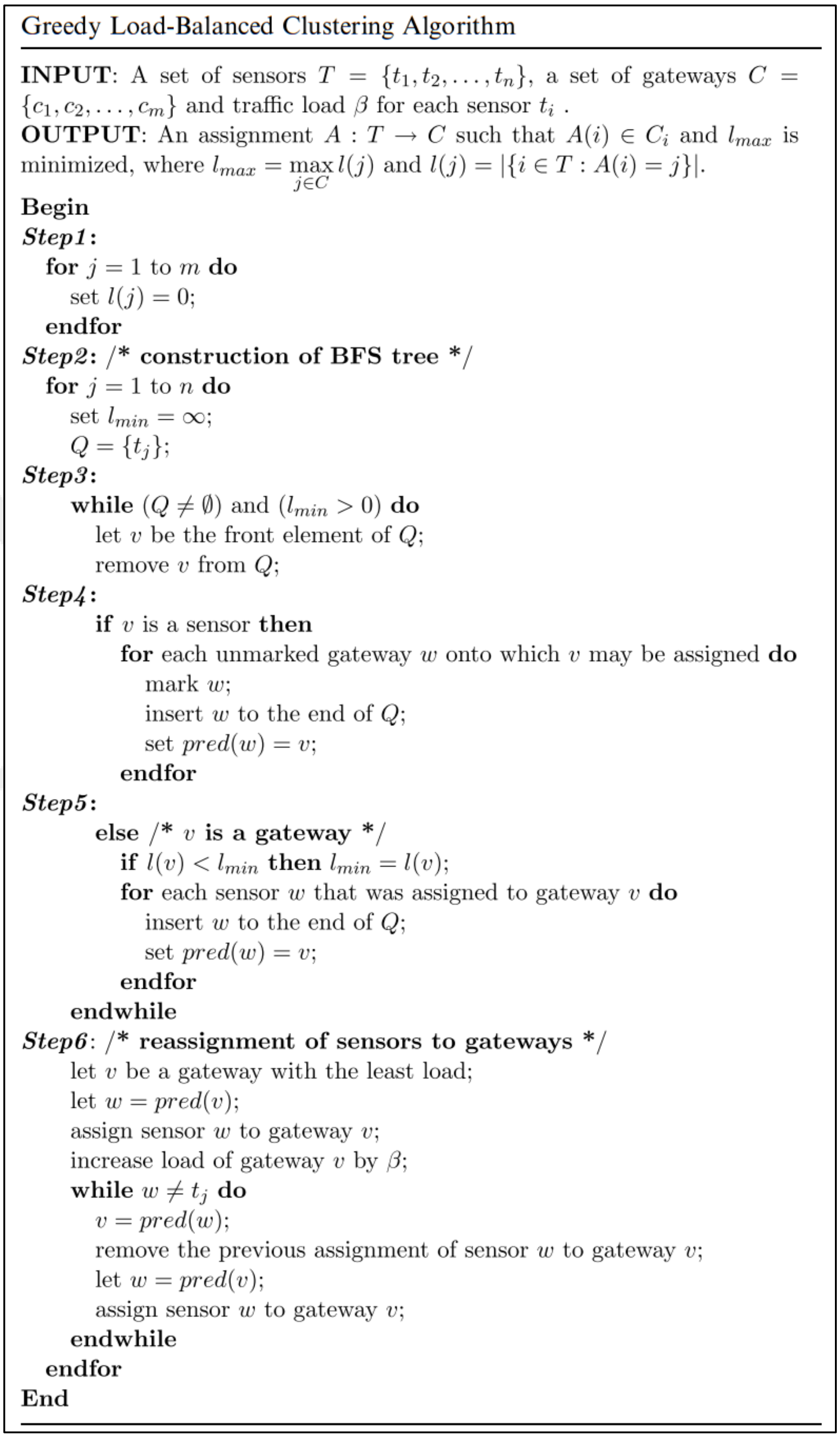
The second stage involves the request of the sink from the Sentinel nodes. This operation is performed when the sink node wants to change its position and at this time the sink node will ask about the amount of energy remaining in that location. This is done by sending a small package to the Sentinel. When this request is made, Sentinel will ask neighboring sensors about their residual energy. The Sentinel node will then collect this information and send it to the sink.

Another protocol for moving a sink proposed in their work assumes that the sink moves uncontrollably and randomly. At each T_{min} the node randomly selects a location to move from within the available D_{max} range. If another location is selected, the sink node will be moved there. This simple approach is known as the Random movement heuristic. In their article, they have been able to achieve this by presenting three solutions to reduce Power usage. They have also been able to improve other parameters that affect network performance, such as overhead, latency.

Disadvantages of the MILP algorithm include its scalability, which has been improved by the introduction of the GREM algorithm. Also, in the RM algorithm, the nodes move unexpectedly, which will reduce the network performance. This algorithm may select the location of sink nodes away from one, which will cause network latency.

[7] proposed a clustering algorithm to increase scalability in sensor networks. They called the proposed clustering algorithm Greedy Load-Balanced Clustering Algorithm (GLBCA). In this algorithm, a set of nodes, called gateway nodes, act as a sink. The suggested technique is meant to distribute traffic among the network's gate nodes evenly. A more stable system and stronger links between nodes in a network are the results of a well-balanced clustering strategy. In this clustering algorithm, it is assumed that the pre-selected gates and the location of the nodes are also specified. Also, to decrease the maximum traffic load on the gate, each node is just a member of a cluster.

The Proposed Approach of their study tries to use the maximum available gateways in order to balance the traffic load in the network. As a result, all nodes share the network's traffic load equally and to do this, first the network nodes will be sorted in ascending order based on the traffic load on them, if the resulting list is represented by T = { t_1, t_2,…, t_n }. Starting from the first node of the t_1 sensor in the sorted list, an attempt is made to assign t_1 to a zero-load gate. The algorithm then tries to assign the t_2 node to another gate with the lowest load in the same way. The algorithm continues in the same way, and in each iteration, an attempt is made to assign a P (augmenting path) to connect the T_i node to a gate with the lowest load. If such a path is found, it amplifies the edges of the P path by assigning T_i to some P nodes and reassigning other sensor nodes to the P path gates. If there is no path that starts with T_i, then we assign T_i to the gate with the lowest load in C_i (least loaded gateway). The algorithm terminates when each node has been paired with a gate. The pseudo-code of the proposed algorithm is shown in figure (1.5).



```
Greedy Load-Balanced Clustering Algorithm

INPUT: A set of sensors $T = \{t_1, t_2, \ldots, t_n\}$, a set of gateways $C = \{c_1, c_2, \ldots, c_m\}$ and traffic load $\beta$ for each sensor $t_i$ .
OUTPUT: An assignment $A : T \to C$ such that $A(i) \in C_i$ and $l_{max}$ is minimized, where $l_{max} = \max_{j \in C} l(j)$ and $l(j) = |\{i \in T : A(i) = j\}|$.
Begin
Step1:
  for $j = 1$ to $m$ do
    set $l(j) = 0$;
  endfor
Step2: /* construction of BFS tree */
  for $j = 1$ to $n$ do
    set $l_{min} = \infty$;
    $Q = \{t_j\}$;
Step3:
    while ($Q \neq \emptyset$) and ($l_{min} > 0$) do
      let $v$ be the front element of $Q$;
      remove $v$ from $Q$;
Step4:
      if $v$ is a sensor then
        for each unmarked gateway $w$ onto which $v$ may be assigned do
          mark $w$;
          insert $w$ to the end of $Q$;
          set $pred(w) = v$;
        endfor
Step5:
      else /* $v$ is a gateway */
        if $l(v) < l_{min}$ then $l_{min} = l(v)$;
        for each sensor $w$ that was assigned to gateway $v$ do
          insert $w$ to the end of $Q$;
          set $pred(w) = v$;
        endfor
    endwhile
Step6: /* reassignment of sensors to gateways */
    let $v$ be a gateway with the least load;
    let $w = pred(v)$;
    assign sensor $w$ to gateway $v$;
    increase load of gateway $v$ by $\beta$;
    while $w \neq t_j$ do
      $v = pred(w)$;
      remove the previous assignment of sensor $w$ to gateway $v$;
      let $w = pred(v)$;
      assign sensor $w$ to gateway $v$;
    endwhile
  endfor
End
```

Figure 1. 5 Pseudo-code of GLBCA clustering algorithm [7]

In this algorithm (GLBCA), it was shown that this problem can be done optimally. It is also shown that if the load on the nodes is not the same, this can be done in exponential time. However, this seems to be NP-Hard (non-deterministic polynomial-time hardness) when the load on the nodes is not the same. The benefits of this algorithm include its

feasibility in exponential time. Also, the proposed approach has been able to distribute the network load on cluster head nodes in a balanced way by using clustering. Disadvantages of this algorithm include that has no make attention to Energy Hole, low Scalability, medium Transfer Delay, high Complexity, and the Energy Aware is rated low for this protocol. The proposed method also uses a centralized approach supposing that every node knows the whole topology of the network. The proposed method is not scalable and does not work well for networks with a large number of nodes.

[8] proposed another clustering algorithm with dual sink and they called this algorithm EEDARS. In this algorithm, there is also a fixed sink and a mobile sink. The authors used an event-based network model to measure how well the suggested algorithm works. In their study, they divided the network into several areas. They then assumed that only one area sensor was sending data at a time. Figure (1.6) shows the EEDARS algorithm with a dual sink.

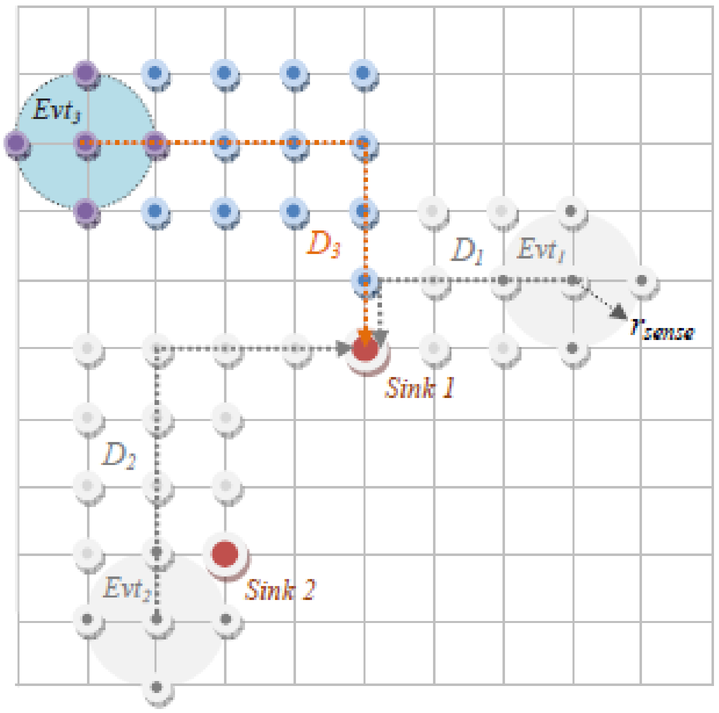


Figure 1. 6 Event-based network model in which Evt3 is currently the active region [8]

They also hypothesized that the mobile sink uses a random motion model. The main idea of this study is to transfer data from the shortest path to the nearest sink. At this time, the sink is moved farther to the center of the network and is considered a fixed sink. The nearest sink also acts as a mobile sink and is as close to the center of the active area as

possible. This prevents all static sink location messages from being transmitted to the sensors. In fact, the proposed algorithm avoids sending additional control messages by shifting the fixed and moving role between the gates. They assumed that each sensor could communicate with a maximum of four neighboring sensors. Therefore, data aggregation in sensors has not been considered. In this study, each sensor selects the closest path to transfer data to the sink. It is also assumed that at any given time, the sensors use GPS to know their position, the position of the sinks, and the position of a hop neighbor. Figure (1.7) shows the path selection in EEDARS algorithm.

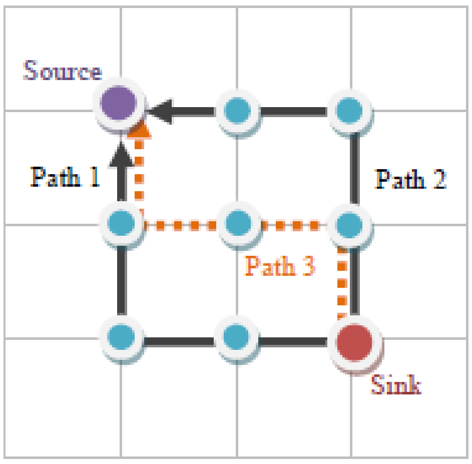


Figure 1. 7 Path selection in EEDARS algorithm [8]

In general, the EEDARS algorithm consisting of two stages:

1- Network setup: In this phase, when the network is in operation, one sink is fixed in the middle, and the other sink wanders about until it finally settles in one specific location. Initially, the sink in the middle of the network is considered a fixed sink and the other mobile sink is considered. Each sensor has a status property that indicates whether the sink is currently stationary or moving. Since the proposed path in this algorithm is the shortest path, each sensor in the network should be aware of the location of the sinks as well as the location of neighboring sensors. According to the fixed sink, it only broadcasts its location and mobile sink to all sensors in the network only once in the start-up phase.

Also in this phase, each sensor is informed of the location of neighboring sensors and creates a table of neighboring sensors.

2- Network function: In this phase, the sensors wait for an event. As soon as an event occurs, the sensors in the event area send their first msg to the fixed sink and wait for a response msg from the fixed sink. While waiting, each sensor in the active area holds all other messages except the first message. Because fixed sinks and mobile sinks can communicate directly with each other, they can share the time it takes to receive the first messages sent from the sensors. Then the fixed sink calculates the average time of sending the message and using equation (1.11), the threshold distance of the sinks from each other is calculated $T_{\mathrm{tx}}^{\mathrm{Evt}}$.

$$R_{\mathrm{threshold}} = \mathrm{v}_{\mathrm{max}} \times T_{\mathrm{tx}}^{\mathrm{Evt}} \qquad (1.11)$$

$\mathrm{R}_{\mathrm{threshold}}$ shows the maximum distance between the sinks. As shown in figure (1.8), the distance of the mobile sink from the fixed sink should not be greater than $\mathrm{R}_{\mathrm{threshold}}$. In fact, $\mathrm{R}_{\mathrm{threshold}}$ guarantees the return time of the mobile sink to the network center in the next event.

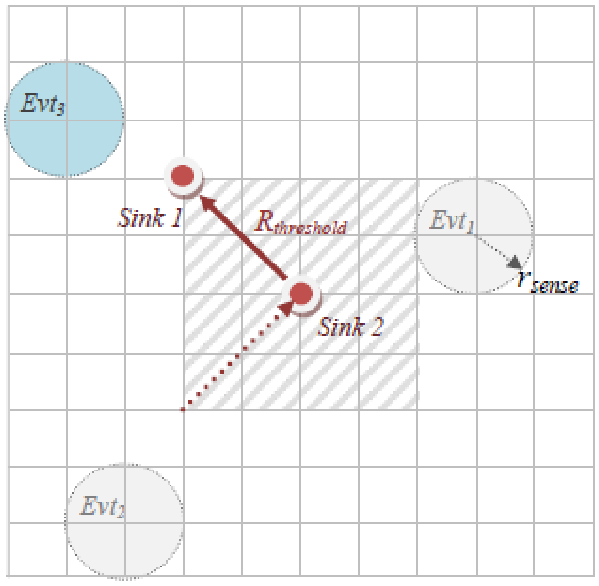
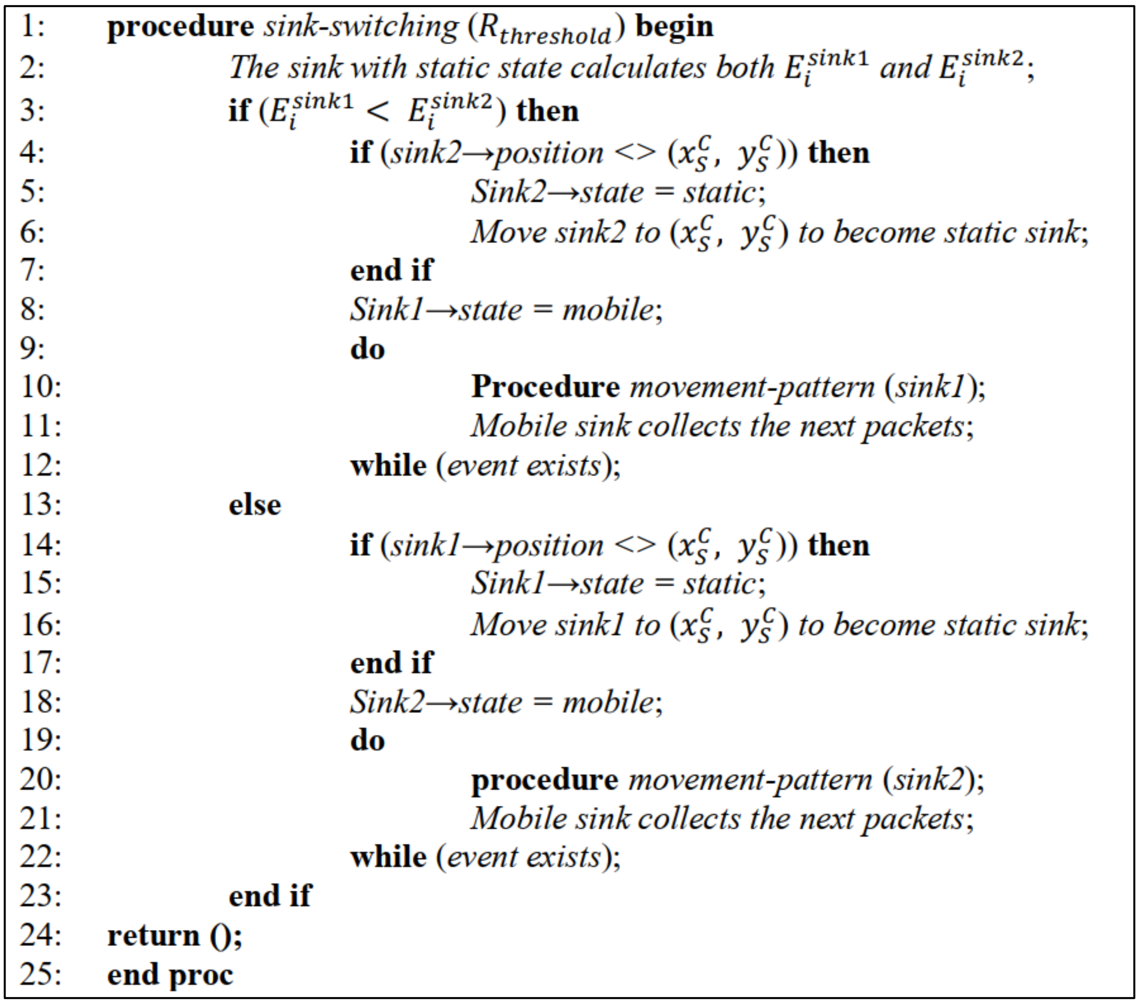


Figure 1. 8 Range of motion of the mobile sink [8]

After the fixed sink calculates the average message sending time, the relocation mechanism executes the role with $\mathrm{R}_{\mathrm{threshold}}$ input data to determine the role of the sinks. Meanwhile, the messages stored in the sensors are also sent to the mobile sink. The pseudocode of the role-shifting mechanism (switching) is shown in figure (1.9).



```
procedure sink-switching ($R_{threshold}$) begin
          The sink with static state calculates both $E_i^{sink1}$ and $E_i^{sink2}$;
          if ($E_i^{sink1} < E_i^{sink2}$) then
                    if (sink2→position <> ($x_S^C$, $y_S^C$)) then
                              Sink2→state = static;
                              Move sink2 to ($x_S^C$, $y_S^C$) to become static sink;
                    end if
                    Sink1→state = mobile;
                    do
                              Procedure movement-pattern (sink1);
                              Mobile sink collects the next packets;
                    while (event exists);
          else
                    if (sink1→position <> ($x_S^C$, $y_S^C$)) then
                              Sink1→state = static;
                              Move sink1 to ($x_S^C$, $y_S^C$) to become static sink;
                    end if
                    Sink2→state = mobile;
                    do
                              procedure movement-pattern (sink2);
                              Mobile sink collects the next packets;
                    while (event exists);
          end if
return ();
end proc
```

Figure 1. 9 Pseudocode of the displacement mechanism between the fixed and mobile role between the sinks [8]

The advantages and disadvantages of the EEDARS routing protocol, which has a high Energy Hole, low Scalability, high Transfer Delay, high Balance in Traffic Load of Cluster Heads, low Complexity, and a centralized Control Pattern. The Energy Aware is rated low for this protocol.

[9] proposed a new algorithm called LEACH-ER to reduce power usage in WSNs. Using this method, they were able to save energy use while increasing the lifespan of the network. This has also improved the network's dependability. In the proposed method, first, an algorithm for selecting a cluster head is introduced that is able to detect energy changes in nodes. Then a concept called service differentiation is introduced that makes it more tolerable to fail. The algorithm also assumes that each sensor node has a data structure in which the pointer field points to the next node. The field shows the identification number of a node. Each node also has a field called ENG, which specifies

the energy of a node. The next field is FLAG, which specifies whether a node can be a head of cluster or not, whose default value is one.

The proposed algorithm of [9], the head of cluster is selected and maintained by the sink. Initially, if the cluster head energy is bigger than one and its value has not changed, the head of cluster delivers the data immediately to the sink. When the head of cluster energy changes to a value greater than one, the cluster head energy decreases by one unit and then delivers the data to the sink. When the head of cluster energy is zero and at the same time it is responsible for sending the packet to the sink node, then it sends a control packet to the sink and announces its exit from the cluster. Then the first node in the list of subsequent nodes whose FLAG value is equal to one is selected as the head of cluster. The current FLAG head of cluster field value will be zero. At the end, the head of cluster sends an all-broadcast message to all the nodes in the cluster and introduces the new head of cluster to them. In the next case, if new nodes are added to the cluster or removed from the cluster, if the nodes' energy does not change, then the nodes send the data packets to the head of cluster and the value of the field ENG_N is a one-bit field. The inside of the package is set to zero. The head of cluster then waits for packets to arrive and checks if the value ENG_N is zero and starts merging data. Otherwise, if the nodes' energy changes, the nodes set the value of ENG_N to one and send the packets to the head of cluster. If the head of cluster detects the value of the received packet ENG_N is one, then it puts the received node ID in a special table and then starts merging the data. After the value of the IDs in the table exceeds the specified threshold value, the head of cluster sends this data to the sink node and clears the data. It then reduces the energy of the corresponding nodes by a value, followed by the sink node updating the list and sorting them according to ENG.

The proposed algorithm uses a mechanism called service diversity when sending data. A variety of services is a protocol for identifying and controlling network traffic based on a particular class, so a particular type of traffic can be a higher priority. A variety of services is the most advanced way to manage traffic. The authors used a variety of services in the proposed approach and in the data transfer phase and were able to improve network reliability.

One of the advantages of this algorithm is the reduction of the number of messages sent between the head of cluster and the members of cluster, which in turn reduces the overhead in the network. The authors also use the service diversification approach to increase network failure tolerance and have been able to greatly improve this criterion. The proposed algorithm does not distribute the load evenly between the cluster head nodes. Its convergence time is also low. In addition, it can be noted that this algorithm is not very stable and is not error resistant.

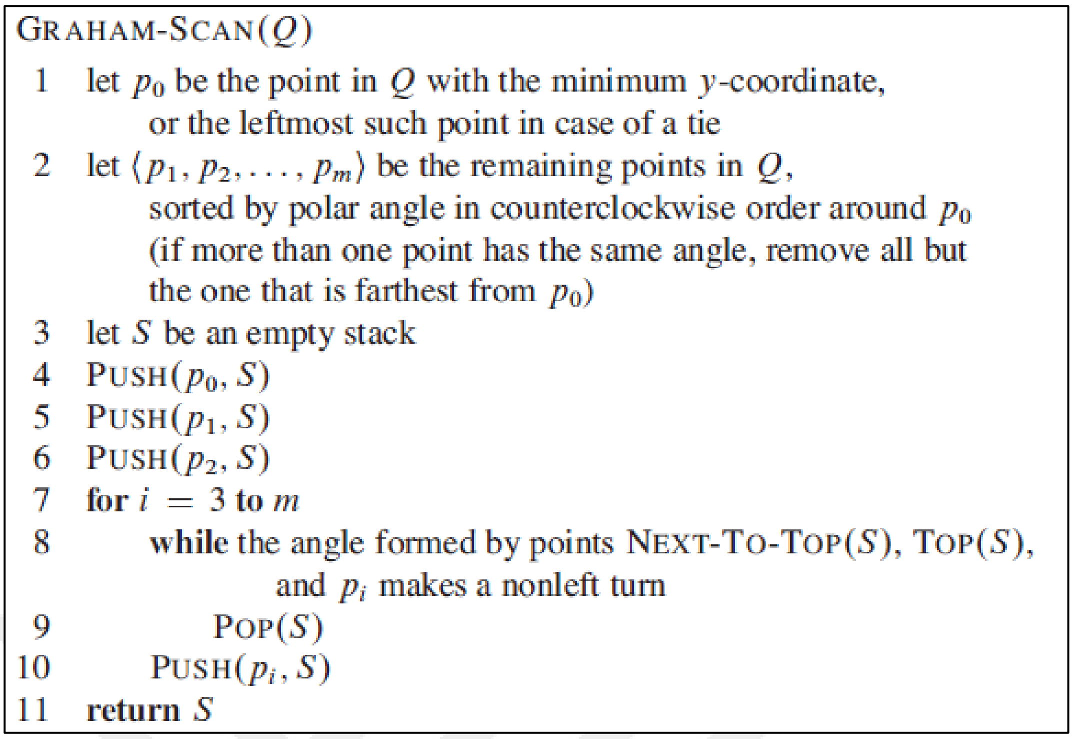
[10] used a convex cover method to discover the proper position of the sink node. The convex coverage of a set of points on the Euclidean plate is the smallest convex set contained in this set. For example, when x is a finite subset of points on a plane, the convex cover may be shown as a bar drawn around x. The proposed algorithm consists of two operational phases of clustering, convex coverage. Using this method, they were able to reduce the distance to the head of cluster. Heads of cluster organize operations such as data aggregation and cluster node management. There are several methods for clustering. The authors use a hierarchical method called LEACH. This method balance the network load by reducing heads of cluster randomly and also reduces energy consumption. Energy consumption for sensor nodes to send data is in equation (1.12).

$$E_S(\mathrm{K,D})=\mathrm{E_{elect}} \times \mathrm{k}+\mathrm{E_{amp}} \times k \times d^2 \tag{1.12}$$

It is also for sending data as an equation (1.13).

$$E_R(k)=\mathrm{E_{elect}} \times k \tag{1.13}$$

In the above equation’s, k is the size of the message sent / received. Also, d is the transition distance. The distance between the transmission and reception of the message has a direct impact on energy consumption. The problem of convex coverage has been used in many fields such as mathematics and computer science. In this paper, the problem of convex coverage is used to discover the position of the hole node. Convex coverage consists of a limited set of points. Several algorithms have been proposed to create convex coverage. In the proposed paper, Graham algorithm is used. Graham's algorithm starts from the lowest point. Creates a simple and closed path by going through the path and increasing the degree. The starting point and the next point should be in one cover. Holds the path and the cover points in two consecutive elements are removed from the beginning of the path. They are then added to the end of the cover sequence and removed. Rotation is used to decide whether to accept or reject the next point. The pseudo-code of Graham algorithm to find the convex cover showing in figure (1.10).



```
GRAHAM-SCAN($Q$)
1   let $p_0$ be the point in $Q$ with the minimum $y$-coordinate,
        or the leftmost such point in case of a tie
2   let $\langle p_1, p_2, \ldots, p_m \rangle$ be the remaining points in $Q$,
        sorted by polar angle in counterclockwise order around $p_0$
        (if more than one point has the same angle, remove all but
        the one that is farthest from $p_0$)
3   let $S$ be an empty stack
4   PUSH($p_0, S$)
5   PUSH($p_1, S$)
6   PUSH($p_2, S$)
7   for $i = 3$ to $m$
8       while the angle formed by points NEXT-TO-TOP($S$), TOP($S$),
                and $p_i$ makes a nonleft turn
9           POP($S$)
10      PUSH($p_i, S$)
11  return $S$
```

Figure 1. 10 Pseudo-code of Graham algorithm to find the convex cover [10]

Using this method, heads of cluster will use less power to send aggregated data to the sink. In previous algorithms, there was a mobile sink. These algorithms boost the lifespan of the network by balancing the power usage of the sensors in the transmission range of the mobile base station. However, every time the sink moves to a new position, it must broadcast the new location to all nodes in the network. Because of this, some mobile sink clustering algorithms assumed that the location of the mobile sink already existed or could be detected by sensors. Therefore, the significant energy consumption of the sensors when receiving and transmitting all-broadcast messages of the sink location was seriously considered.

In the table (1.1) below, all the algorithms and methods that were explained in the chapter are compared.

Table 1.1 Evaluates the routing protocols examined in terms of various parameters

Protocol Name	Energy Aware	Control Pattern	Complexity	Balance In the Traffic Load of Cluster Heads	Transfer Delay	Scalability	Pay Attenti-on to The Energy Hole
EEUC	High	Distributed	High	Medium	Medium	High	Yes
DS	High	Distributed	Medium	Medium	Medium	High	Yes
PSO	High	Distributed	Medium	Medium	Medium	Medium	No
GAbEERP	High	Distributed	High	High	Low	High	No
GREM	Low	Centralized	Medium	Low	Medium	Low	Yes
GLBCA	Low	Centralized	High	High	Medium	Low	No
EEDARS	Low	Centralized	Low	High	High	Low	Yes
LEACH-ER	High	Centralized	High	High	Low	High	No
GRAHAM-CONVEX	High	Centralized	Low	High	Medium	High	No

The table (1.1) provides a comparison of different routing protocols in terms of several parameters. These parameters include Energy Hole, Scalability, Transfer Delay, Balance in Traffic Load of Cluster Heads, Complexity, Control Pattern, Energy Aware, and Protocol Name. From table (1.1), we can conclude that different routing protocols have varying strengths and weaknesses in terms of the evaluated parameters. For example, protocols such as EEUC, DS, and GAbEERP are energy-aware and suitable for scenarios where energy conservation is critical. On the other hand, protocols like GREM and GLBCA are efficient in terms of transfer delay, while EEDARS is better suited for maintaining a balanced traffic load of cluster heads. It is also evident that centralized protocols generally have lower complexity and control pattern, while distributed protocols can be more scalable. Furthermore, the table indicates that some protocols, such as LEACH-ER and GRAHAM-CONVEX, are not suitable for scenarios with energy holes, while others, such as PSO and GLBCA, are not energy-aware. Therefore, the selection of a routing protocol should be based on the specific requirements and characteristics of the WSNs application in question.

In this research, an extensive evaluation is presented to show the efficiency of the proposed algorithm. The authors in this study proved that the proposed algorithm has a longer lifetime than the PSO algorithm. Also, standard deviation in energy consumption and transmission delay has decreased and the delivery rate has increased. However, in this study, energy holes [11] have not been investigated. Not paying attention to the location of nodes in clustering causes hot spots. Hotspots are cluster heads from the network that are close to the sink or on busy interstitial routes. Networks where there are hotspots suffer from the energy hole problem [11]. An energy hole is the premature death of nodes around a WSNs that prevents data from being transferred to the sink. In this research, unbalanced clustering technique has been used to solve this problem. In this technique, the size of the clusters is calculated according to the distance of the cluster head from the sink. In this technique, cluster heads that have close members have fewer members than clusters far from the sink. Another technique to solve the hole problem is to use a mobile sink. In this technique, the sink is moving to places where the density of living nodes is higher. This technique increases the balance in data transmission over the WSNs nodes. The use of either technique reduces the energy hole problem as much as possible. In this research, unbalanced clustering technique has been used to prevent energy holes [11].

1.2 Objective of Thesis

Due to the fact that battery technology has not kept up with electronics and communications, WSN nodes provide a significant challenge in terms of power supply [12]. However, wireless sensor networks are predicted to run for years without needing a change of energy source because to the large number of nodes, which are hard to reach and cannot be changed [13]. Longevity enhancement in the network is the primary focus of this research and this investigation on sensor power usage is an effort to shed light on this pressing topic and in this study, we took into account not just the energy needs of the cluster heads, but also the energy needs of the cluster members [12]. Also, the routing protocol based on the MFO Algorithm have taken the problem of energy holes into account [11]. So, to provide a cluster-based routing by using MFO Algorithm to extend the life of the WSNs, we will use the following methods:

- Using Moth-Flame Algorithm in both clustering and routing phases [14].
- Using unbalanced clustering techniques to avoid energy holes [11].
- Using the reliability parameter in the routing utility function to balance the power usage of the middle clusters of the data transfer path to the sink.
- Taking into account how far away cluster members are from the cluster heads to optimize energy consumption in cluster members.

1.3 Hypothesis

- Wireless sensors have limited computational and energy resources.
- The sink is connected to a power supply, so there are no computational and energy constraints.
- The sink can be inside or outside of network.
- Sink and sensors are fixed.
- Cluster sensors send information received from the environment to the cluster head. The network's death was anticipated to occur when the first sensor died.

1.4 Description and Expression of the Problem

As a rapidly evolving field, WSNs have many potential uses, including but not limited to infrastructure security, technological sensing, environmental control, information computing, and more [15]. These networks are made up of a huge number of individual nodes that communicate with one another wirelessly [16]. A WSNs is made up of sensor nodes, sink, and monitored events, so power supply, network interface, processing unit, and sensing unit are the four main parts of a sensor node [17]. The sensory part takes readings from the environment and interprets them, whereas the processing part measures some other physical property and data is sent and received wirelessly between nodes and back to the base station or end user, the power unit is a compact source of energy for all the other parts and the monitored event's execution mode might be either dynamic or static [18].

There are significant difficulties in WSNs due to the energy constraints of sensor batteries and their inability to be recharged where sensors in WSNs are generally put in harsh situations where they cannot be readily managed, also transmission of data is a major contributor to WSNs' energy needs, so in order to maximize efficiency and reduce transmission overhead, WSNs often form clusters [19]. Each node in a clustered WSNs reports to a single node that serves as the cluster head, it is the responsibility of each node in a sensor network to monitor for events, process data effectively, and relay that information to other nodes, as a consequence, there are three distinct components to WSNs' total energy footprint: the energy used by the sensor node during environmental data collection, the energy used by the sensors and cluster head to analyze that data, and the energy used to transmit that processed data to the base station [19]. In figure (1.11) example of clustering in WSNs [19].

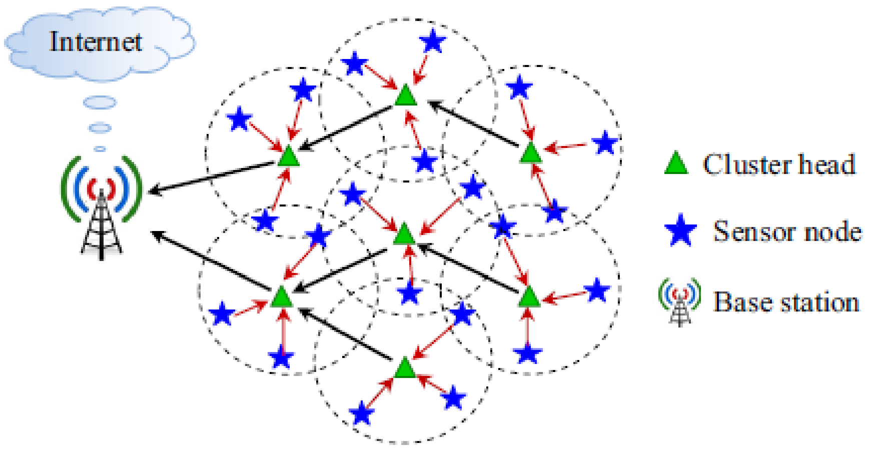


Figure 1. 11 Clustering in WSNs [19]

Noise levels, for example, might reduce the quality of the perceived signals and cause the sensor to use more power than necessary, however this varies depending on the application [12]. It has been noticed that communication often uses more energy than calculation, and that processing data requires less energy than wireless transmission [20]. Data packet transmission and reception use about the same amount of power in close-range connections [21]. The stability of the network may be maintained for a longer period of time if the load of traffic is spread out equally throughout the network [22]. Transmission of a piece of data often consumes several thousand times more energy than processing the same quantity of data, so communications should be a top priority when selecting a method for data transfer, along with other factors like the network's requirements and the sensor's power consumption [23]. So, to solve this problem, clustering is a commonly used technique to reduce energy consumption and improve network performance, which is the focus of this study.

1.5 Importance and Necessity of Research

Reducing power consumption in WSNs is a major motivation for our study of cluster-based routing protocols implemented by using the MFO Meta-Heuristic Algorithm. The huge number of nodes in WSNs, all of which have their own energy limits, makes this problem of energy efficiency paramount in the design of WSNs. This study's objective is to find a solution to this problem by enhancing WSN routing efficiency using a MFO meta-heuristic method. We anticipate that by dispersing the energy load throughout the network, the clustering component of our strategy will further decrease energy consumption and extend the lifespan of the network. With the potential to greatly affect the design and deployment of WSNs for applications as diverse as environmental control, commercial sensing, and security management, this study is crucial to the development of more energy-efficient WSNs.

1.6 Research Method

In general, the research method will be as follows:

- Defining the exact problem and defining the steps to be taken.
- Design of a cluster-based routing algorithm using Moth-flame Algorithm.
- Simulation the cluster-based routing MFO Algorithm and routing PSO Algorithm in OMNET ++.
- Examine the efficiency of the suggested cluster-based routing method and outline its proper use.
- Analyze the improvements made by the suggested routing algorithm compared to previous routing method that relied on clusters.

The present dissertation is prepared in 4 chapters and thus. In Chapter 2, overview of MFO algorithm presented. In Chapter 3, proposed method by the Moth-Flame Optimization (MFO) Meta-Heuristic Algorithm is presented. In Chapter 4, the efficiency of the proposed protocol is compared to a number of protocols published in the literature, according to the criteria of energy consumption evaluation, justice in cluster energy consumption and network life. Finally, presents conclusions and future work.

2
OVERVIEW OF MFO ALGORITHM

2.1 Introduction

This chapter examines some cluster-based routing protocols in WSNs. In some of these protocols, in addition to consuming clusters' head energy the energy consumption of cluster members has also been considered.

2.2 Moth-Flame Optimization Algorithm

2.2.1 The Behavior of The Moth to Identify the Path

The most important fact in the moths is their movement at night, they fly at night using the moonlight and they use a transverse direction to move at night [14]. Moths using this mechanism fly at a constant angle to the moon, making it ideal for long-distance travel in a straight line [24]. Since the moon is too far away for the moths to fly directly to it, this mechanism ensures that they will fly in a straight path [14]. An example of this kind of flight is seen in figure (2.1).

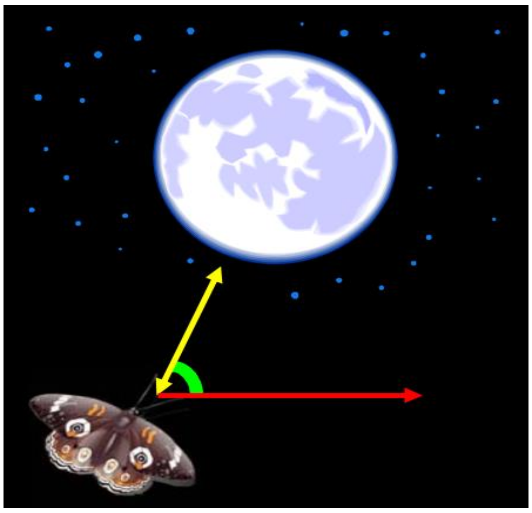


Figure 2. 1 The transverse direction of the moth relative to the moon [14]

Apart from this moth behavior, we see that the moths move around the smart lights, this is due to the inefficiency of the transverse direction when the light source is close to the moth [25]. The moth initially tries to find a transverse direction to the lights, since the light is very close to the moth than the moon, the moth chooses a deadly spiral path to reach the light [24]. In figure (2.2) shows a moth spiral movement [14].

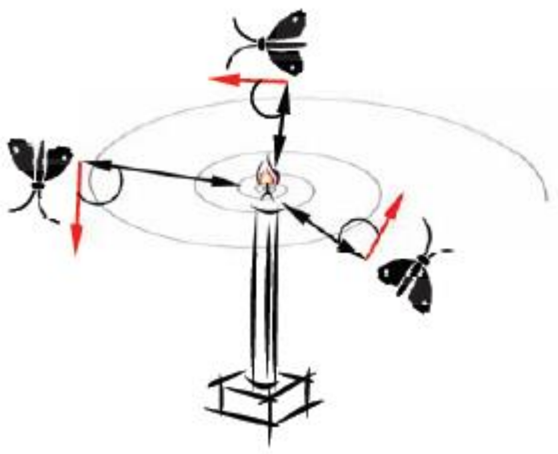


Figure 2. 2 Moth spiral movement [14]

As shown in figure (2.2), the moth eventually reaches the light, and the following is a mathematical model for this behavior of the moth given in Section 2.2.2.

2.2.2 Mathematical Modeling of MFO Algorithm Behavior

The algorithm works under the premise that the moths are possible candidates, and that the only independent variable is their geographical position, hence, moths are capable of moving in a single, double, or even higher dimensional space [25, 26]. Since the MFO is a meta-heuristic algorithm, the set of moths is an array of (n*d), equation (2.1) shows the set of moth's sets [14].

$$M=\begin{bmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,d} \\ M_{2,1} & M_{2,2} & \cdots & M_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ M_{n,1} & M_{n,2} & \cdots & M_{n,d} \end{bmatrix} \quad (2.1)$$

Where (n) corresponds to the total number of moths, (d) number of search space dimensions for optimal response, (M) matrix of Moth collection and ($M_{n,d}$) the number of moths in which dimension for optimal response, Also, in this algorithm a n-vector is defined for the use of the moths. In equation (2.2) shows the usefulness of the moths, where (n) corresponds to the total number of moths and (Om_n) the value of the usefulness of the n moths.

$$OM=\begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \tag{2.2}$$

Another key concept in the MFO algorithm is flames. The set of flames is like a moths set as a matrix of (n*d) where (n) corresponds to the number of flames, (d) is the number of dimensions of the search space for optimal response and ($F_{n,d}$) the number of flames in which dimension for optimal response. In equation (2.3) shows the set of flame sets [27].

$$F=\begin{bmatrix} F_{1,1} & F_{1,2} & \dots & F_{1,d} \\ F_{2,1} & F_{2,2} & \dots & F_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ F_{n,1} & F_{n,2} & \dots & F_{n,d} \end{bmatrix} \tag{2.3}$$

Also, in this algorithm a n-vector is defined for the usefulness of the flames. Equation (2.4) shows the usefulness of flames where (n) corresponds to the number of flames and (OF_n) shows the value of the usefulness of n flame.

$$OF=\begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix} \tag{2.4}$$

It's important to remember that both moths and flames are potential candidates, the only discernible difference lies in their respective behaviors and update schedules [27]. moths are real search agents that search the problem-solving space, while flame is the best moths ever obtained, in this algorithm, each moth moves around a flame, and when one finds a better candidate, it updates the new candidate with a recent flame and this method ensures that a moth will always have access to the optimal answer [27].

The moth-flame optimization algorithm is a triple that is approximately a global optimal response to enhancement problems. In equation (2.5) Mathematical Model shows the moth-flame optimization algorithm [14].

$$MFO=(I,P,T) \tag{2.5}$$

In the equation (2.5), (I) is a function that randomly selects a set of initial candidate answers. This set is considered as a set of moths. The scientific model of the function I is defined in the Equation (2.6).

$$\mathrm{I} : \phi \rightarrow \{\mathrm{M},\mathrm{OM}\} \tag{2.6}$$

The algorithm's central function is denoted by P, it is this function that is responsible for relocating the moths throughout the search space. This function receives an M matrix at each step and the location of the moths in that matrix updates and then returns the updated Matrix M, the scientific model of the function P is shown in the (2.7) [28].

$$\mathrm{P} : \mathrm{M} \rightarrow \mathrm{M} \tag{2.7}$$

The T function examines the end criterion of the algorithm, the function returns True if the desired outcome has been reached and False otherwise, the scientific model of function T is shown in the Equation (2.8) [28].

$$\mathrm{T} : \mathrm{M} \rightarrow \{\mathrm{true},\mathrm{false}\} \tag{2.8}$$

The algorithm also defines the two LB and UB double vectors, which show the lower and upper borders of the moth location. Equations (2.9) and (2.10) show LB and UB vectors in which $\mathrm{LB}_i$and UB_i are lower and upper bounds on the location of the moth i.

$$\mathrm{LB} = \{\mathrm{LB}_1, \mathrm{LB}_2, \ldots, \mathrm{LB}_i\} \tag{2.9}$$

$$\mathrm{UB} = \{\mathrm{UB}_1, \mathrm{UB}_2, \ldots, \mathrm{UB}_i\} \tag{2.10}$$

As mentioned earlier, the Moth-Flame Algorithm has used the moth behavior towards the flame called the transverse direction. Equation (2.11) shows the mathematical model of the moth behavior in which the M_i moth updates its location according to F_j. The S function also creates a coil movement for the moth, any coil function can be used in this algorithm. Equation (2.12) shows the proposed coil function [29].

$$M_i = \mathrm{S}(M_i, F_j) \tag{2.11}$$

$$S(M_i, F_j) = \cos(2\pi \mathrm{t}) + \mathrm{F}_j . D_i . e^{\mathrm{b\,t}} \tag{2.12}$$

In the equation (2.12), D_i is the Euclidean (**i**th) moth from (**j**th) flame and (b) is a fixed number for coil movement. In this study: b = 1 is considered. (t) is also a random digit in the closed interval [1,1-]. Equations (2.13) and (2.14) obtain a random digit in the t interval [14].

$$\mathrm{a} = -1 - \frac{\mathrm{Iteration}}{\mathrm{Max\_iteration}} \tag{2.13}$$

$$t = ((a-1) \times \text{rand})+1 \qquad (2.14)$$

Equation (2.15) Euclidean distance between **i**th moths and **i**th flames.

$$D(\text{i,j}) = |M_{\text{i,j}} - F_{\text{i,j}}| \qquad (2.15)$$

The main part of the MFO Algorithm is the coil of the moth around the flame. However, if the moths' sizes are proportional to the number of flames, the optimization process time will increase. For this purpose, an adaptable mechanism is presented to decrease the number of flames, at each step, the size of the Flame_no from the arranged set of moths is considered as a set of flames. Equation (2.16) acquires the value of Flame_no at each stage [30].

$$\text{Flame\_no} = \text{round}(\frac{\text{N-1}}{T} \times \text{N-L}) \qquad (2.16)$$

Equation (2.16) displays the number of flames generated at each step of the algorithm's execution. L and T show the number of recent execution steps and the maximum stage of execution of the MFO algorithm. N also shows the maximum number of flames. Figure (2.3) shows a reduce in the number of flames on the number of algorithm performance [14]. Moth-Flame optimization algorithm flowchart and Pseudo-Code is shown in figure (2.4) and (2.5).

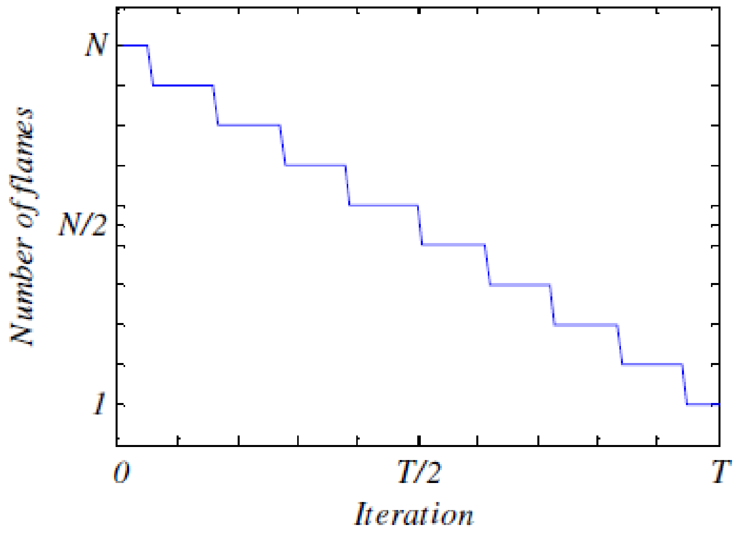


Figure 2. 3 Minimize flames while maximizing algorithm implementations [14]

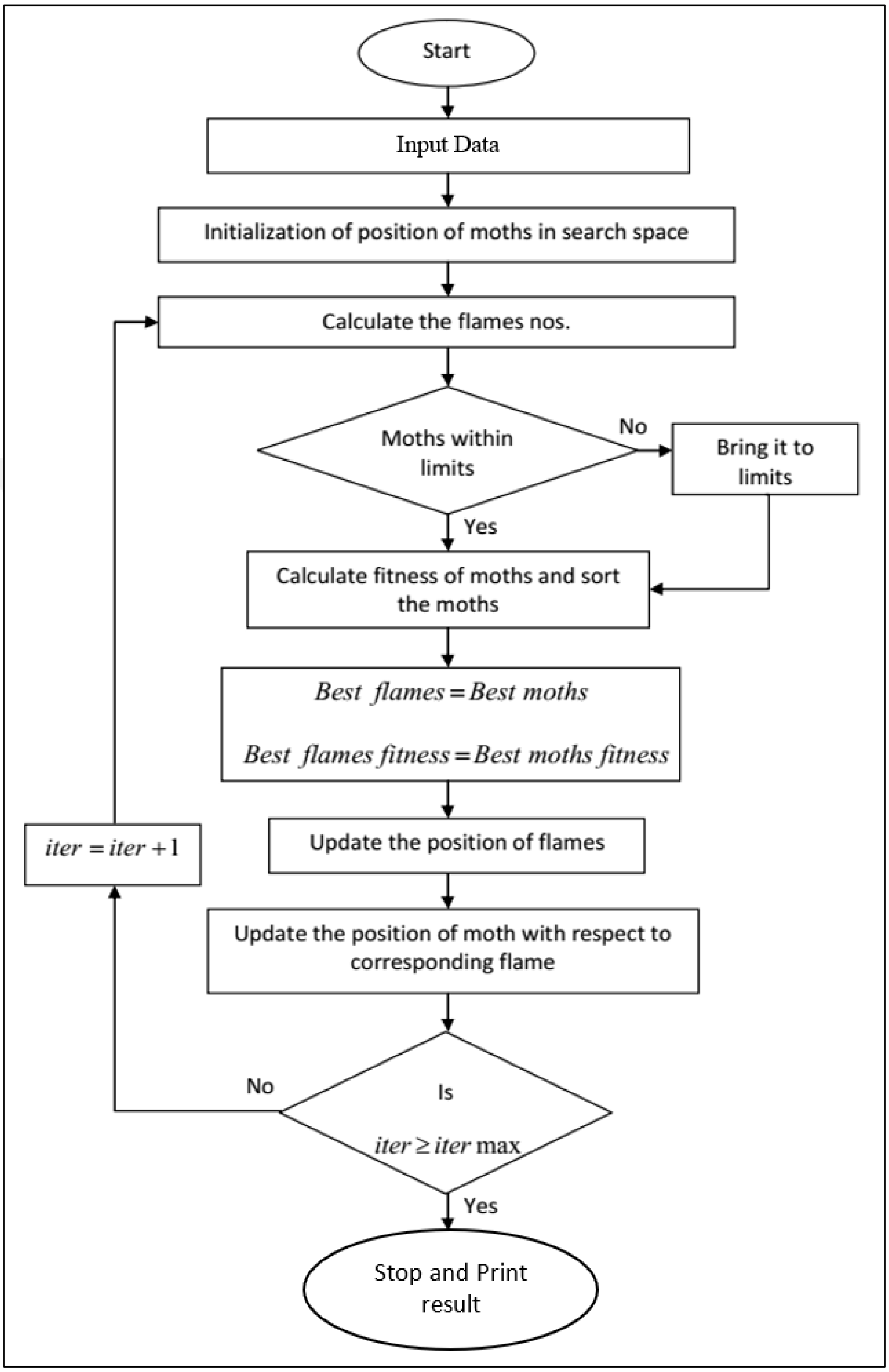
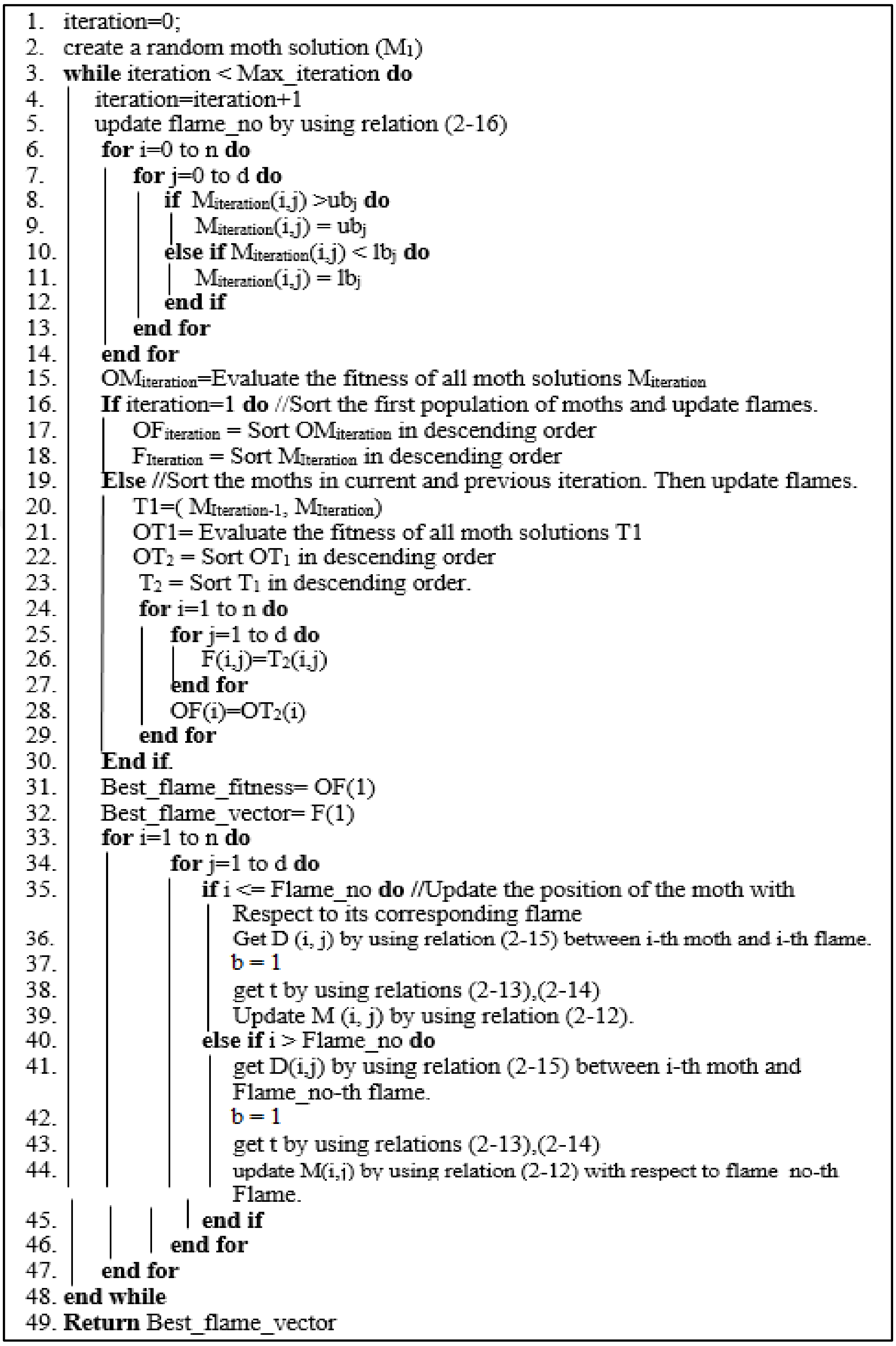

Start
Input Data
Initialization of position of moths in search space
Calculate the flames nos.
Moths within limits
No
Bring it to limits
Yes
Calculate fitness of moths and sort the moths
$Best\ flames = Best\ moths$
$Best\ flames\ fitness = Best\ moths\ fitness$
Update the position of flames
Update the position of moth with respect to corresponding flame
$iter = iter + 1$
No
Is
$iter \geq iter\ \max$
Yes
Stop and Print result

Figure 2. 4 Flowchart of MFO algorithm [14]



```
1.  iteration=0;
2.  create a random moth solution ($M_1$)
3.  while iteration < Max_iteration do
4.      iteration=iteration+1
5.      update flame_no by using relation (2-16)
6.      for i=0 to n do
7.          for j=0 to d do
8.              if $M_{iteration}(i,j)$ >$ub_j$ do
9.                  $M_{iteration}(i,j) = ub_j$
10.             else if $M_{iteration}(i,j) < lb_j$ do
11.                 $M_{iteration}(i,j) = lb_j$
12.             end if
13.         end for
14.     end for
15.     $OM_{iteration}$=Evaluate the fitness of all moth solutions $M_{iteration}$
16.     If iteration=1 do //Sort the first population of moths and update flames.
17.         $OF_{iteration}$ = Sort $OM_{iteration}$ in descending order
18.         $F_{Iteration}$ = Sort $M_{Iteration}$ in descending order
19.     Else //Sort the moths in current and previous iteration. Then update flames.
20.         T1=( $M_{Iteration-1}$, $M_{Iteration}$)
21.         OT1= Evaluate the fitness of all moth solutions T1
22.         $OT_2$ = Sort $OT_1$ in descending order
23.         $T_2$ = Sort $T_1$ in descending order.
24.         for i=1 to n do
25.             for j=1 to d do
26.                 F(i,j)=$T_2$(i,j)
27.             end for
28.             OF(i)=$OT_2$(i)
29.         end for
30.     End if.
31.     Best_flame_fitness= OF(1)
32.     Best_flame_vector= F(1)
33.     for i=1 to n do
34.         for j=1 to d do
35.             if i <= Flame_no do //Update the position of the moth with
                    Respect to its corresponding flame
36.                 Get D (i, j) by using relation (2-15) between i-th moth and i-th flame.
37.                 b = 1
38.                 get t by using relations (2-13),(2-14)
39.                 Update M (i, j) by using relation (2-12).
40.             else if i > Flame_no do
41.                 get D(i,j) by using relation (2-15) between i-th moth and
                    Flame_no-th flame.
42.                 b = 1
43.                 get t by using relations (2-13),(2-14)
44.                 update M(i,j) by using relation (2-12) with respect to flame  no-th
                    Flame.
45.             end if
46.         end for
47.     end for
48. end while
49. Return Best_flame_vector
```

Figure 2. 5 Pseudo-Code of MFO algorithm [14]

3
PROPOSED METHOD

3.1 Introduction

A Wireless Sensor Network is a type of network that consists of spatially distributed autonomous devices that use sensors to monitor physical or environmental conditions. WSNs are used in a variety of applications such as monitoring and controlling industrial processes, home automation, traffic control, surveillance, and so on. Routing is one of the most challenging problems in WSNs due to the limited resources available in the nodes. Energy efficiency is an important factor when designing routing protocols for WSNs since it directly affects the lifetime of the network. Therefore, routing protocols must be designed to reduce energy consumption while maintaining high performance and reliability, so in this chapter, a cluster-based routing protocol by using MFO algorithm will be presented, and this protocol consists of two operational phases:

1. Clustering
2. Routing.

In this research, I will propose a mechanism for the clustering phase using the Moth-Flame optimization algorithm [26]. In the utility function of the cluster head, in addition to the distance of the cluster head from the sink, the distance of the cluster members from the cluster head, the number of cluster members and the remaining energy of the cluster head are also included. In the clustering phase, each sensor in the network selects a (gi) gateway as the cluster head [31]. At the end of the clustering phase, network sensors are divided into several clusters [32]. Cluster heads are a set of gateways and are responsible for collecting, condensing and sending the collected data to the sink [33]. Also, in the routing phase, I design a mechanism using the Moth-Flame optimization algorithm [25]. In the routing utility function, the number of steps and the maximum route transfer distance are considered [34]. The purpose of this work is to solve the contradiction between the number of steps and the maximum transmission distance in order to increase the lifetime of the network.

3.2 System model and specialized terms:

In this research, we used the Moth-Flame optimization algorithm (MFO) to provide a clustering mechanism to improve the cluster-based routing protocol and in the following, the network model, energy model and specialized terms are explained.

3.2.1 Energy model:

In this research, a simple radio model is considered for the energy dissipation of radio hardware. As shown in figure (3.1), the radio energy loss in this model includes the energy required for radio transmission, the energy required for the signal amplifier and the energy required for receiving data.

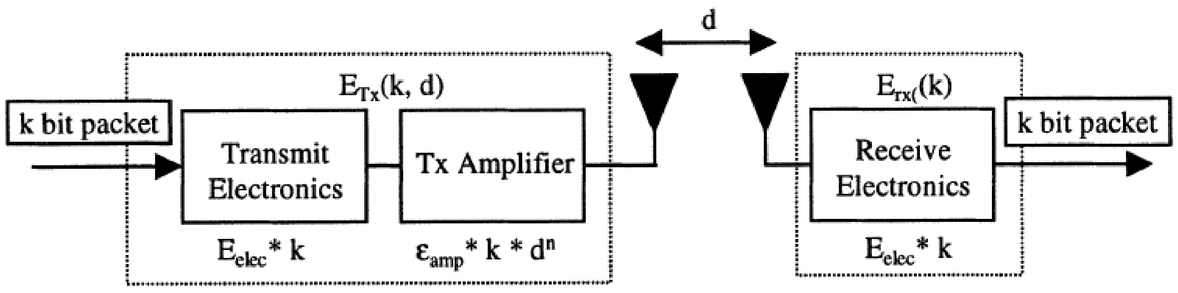


Figure 3. 1 Radio energy loss for sending k message bits [21]

In this radio model, depending on the distance between the transmitter and the receiver, both the free space channel transmission model (d^2 signal loss) and the multipath fading channel transmission model (d^4 signal loss) are used [21]. The use of these two models with a suitable signal control setting in the signal amplifier is provided. We assume that in this network, if the distance between the transmitter node and the receiver node is less than the threshold d_0, the free space transmission model is used, and otherwise, the multipath fading transmission model is used [35]. According to what was said, the energy consumption for sending and receiving L message bits in the distance d is obtained by using equations (3.1) and (3.2) respectively [36].

$$E_{TX}(L,d)=E_{TX\text{-}elec}(L)+E_{Tx\text{-}amp}(L,d)$$
$$E_{TX}(L,d)=\begin{cases} LE_{elec}+L\varepsilon_{fs}d^2, & d<d_0 \\ LE_{elec}+L\varepsilon_{mp}d^4, & d\geq d_0 \end{cases} \tag{3.1}$$

$$E_{RX}(L)=E_{RX\text{-}elec}(L)=LE_{elec} \tag{3.2}$$

E_{TX} and E_{RX} indicate the energy required to send and receive the message, respectively [37]. $E_{\text{TX-elec}}(L)$=LE_{elec} and $E_{\text{RX-elec}}(L)$=LE_{elec} it shows the initial energy required to send and receive L bits of the message, respectively. ε_{fs} and ε_{mp} the energy required for signal amplification is shown in free space channel transmission and multipath fading models, respectively. The initial energy of E_{elec} depends on various factors such as digital encoding, modulation and filtering and dispersion [38].

3.2.2 Network model:

In our model, we have deployed the gateways in a mesh topology and sensors are also randomly distributed in the problem-solving space because in a mesh topology, nodes are connected to each other in a way that allows for multiple paths between them, which provides redundancy and helps to ensure that the network remains connected even if some nodes fail and this can be useful in a WSNs where communication paths may be disrupted due to environmental factors or node failures [39]. We also assume that the sink is outside the gateway mesh structure and all communication in this network model is done wirelessly. A sensor can be assigned to any gateway within its transmission range. Therefore, each sensor has a list of gateways that are in its transmission range [40]. However, each sensor can only choose one of the gates to transmit information. Data collection in the proposed protocol is performed in turn, like the LEACH protocol [41]. At each turn, sensors collect local data and send it to their cluster heads, the cluster head is the gate assigned to the sensor in the clustering phase [42]. Then, after a period of time, each cluster head condenses the data received from its member's sensors and sends it to the sink through other gateways [43]. In addition to gateways, cluster heads are also present in the routing process of data transfer from the cluster head to the sink. In order to reduce energy consumption, the sensors turn off their radios between two consecutive time periods [44]. In this network, when two nodes are in the transmission range of each other, a wireless link is considered.

In this research, MAC layer type TDMA is used to establish communication between intermediate gateways as well as between sensors and cluster heads [45]. In addition, we assume that the gateways use the MAC layer type CSMA/CA to discover the sink and communicate with it [46]. According to what was said, there are two radios in each gateway or cluster head, the first one is for communication with other gateways and the second one is for communication with the sink, Therefore, the first radio is of TDMA type and the second radio is of CSMA/CA type [47]. The authors provided three definitions for network lifetime [48]. Some of them consider the lifetime of the network as the time when the first node dies, others consider the time when the last node dies, and some consider the time when a favorable percentage of nodes die [49]. In addition, the researchers considered the lifetime of the network as the time it takes for data collection to fail in an area [50].

3.2.3 Symbols and specialized terms:

In this research, we use the following symbols:

• **Sensor:** A device that detects the occurrence of a situation or the value of a physical quantity and turns it into an electrical signal and there are different types of sensors such as temperature, pressure, humidity, light, accelerometer and magnetometer [51].

• The set of sensors in a wireless sensor network is denoted by **S=** $\boldsymbol{\{s_1, s_2 \ldots, s_n\}}$, where n is the number of sensors in the network.

• **Gateway:** responsible for collecting, condensing and sending the collected data to the sink [52].

• **Sink:** A node that is responsible for data collection and communication between sensor nodes [53].

• The set of cluster heads in the wireless sensor network is represented by **CH=** $\boldsymbol{\{CH_1, CH_2 \ldots, CH_V\}}$, where v is the number of cluster heads after running the clustering algorithm.

• **Candidate answer (i):** It is a matrix, the number of houses in this matrix is equal to the number of sensors in the clustering phase and equal to the number of gateways in the routing phase, where the sensor must choose the gateway as the cluster head in the clustering phase. And the gateway that must choose another gateway as the next hub [25].

• **Moths and Flames:** both are candidate answers. The only difference between them is in their behavior and how they are updated. Moths are real search agents that search the problem-solving space, while flames are the best Moths ever made [26].

• j = gateway, g = cluster head.

• gj = gateway that is cluster head.

• **Member-Number (gj):** the number of sensors that have selected gateway (gj) as the cluster head [54].

• **L (i):** shows the lifetime of the gate g_i (cluster head g in the answer of candidate i) at the moment [55]. Assuming $E_{\text{residual}}(g_i)$ as the remaining energy and $E_{\text{Gateway}}(g_i)$ as the energy consumed in each turn by the gateway $g_i$we have:

$$L(i) = \left\lfloor \frac{E_{\text{residual}}(g_i)}{E_{\text{Gateway}}(g_i)} \right\rfloor \tag{3.3}$$

• **N_0:** The threshold limit shows the number of cluster head members that can choose a gateway as the cluster head [56]. The higher the number of cluster head members, the lower the failure tolerance.

• **R_{gmax}:** shows the maximum range of information transmission in gateways [57].

• **R_{smax}:** shows the maximum range of information transmission in sensors [58].

• **GoSS (s_i) (Gateways of Sensor Set):** shows the set of gateways that sensor s in candidate solution i is in their competitive radius range and also, they are in the transmission radius range of the sensor [3]. Equation (3.4) obtains the set of GoSS gates for sensor s_i.

$$\text{GoSS}(s_i) = \{g_j | \text{dis}(s_i, g_j) \leq R_{\text{smax}} \cap \text{dis}(s_i, g_j) \leq \text{RC}(j),\, g_j \in G\} \tag{3.4}$$

Equation (3.5) shows the competitive radius: the j-th gate used in the unbalanced clustering technique [3]. g_{farthest} is the farthest gateway from the sink. RC represents the transmission radius of gate g_j. R_{gmax} indicates the transmission range of the gate.

$$\text{RC}(g_j) = \left(\frac{\text{dis}(g_j, \text{BS})}{\text{dis}(g_{\text{farthest}}, \text{BS})}\right) \times R_{\text{gmax}} \tag{3.5}$$

• **GoGS (g_j) (Gateways of Gateway Set):** shows the set of gates that are in the transmission range of the sensor [3].

• **PNextHops(g_i):** a set of gateways that are in the transmission range of the g_i gateway and can be selected as the next node in the path of data transmission to the sink [3].

• **NextHop(g_i):** is a gateway from the PNextHops(g_i) set, which is selected as the next node in the path of information transmission from gateway g_i to sink g_{m+1}. If sink g_{m+1} is in the range of g_i information transmission, g_i considers sink g_{m+1} as the next node [3].

• **HopCount(g_i):** shows the number of intermediate gateways in the path of data transmission from gateway g_i to sink [3]. If the sink gateway is in the transmission range of the g_i gateway, HopCount(g_i)=1. The method of calculating HopCount is shown in equation (3.6).

$$\text{HopCount}=\begin{cases} 1, & \text{NextHop}(g_i) = g_{m+1}\,(g_{m+1} \text{ is sink}) \\ 1 + \text{HopCount}(g_i), & \text{NextHop}(g_i) = g_j\,(g_j \text{ is not sink,} \\ & \text{but is nearer to sink than } g_i) \end{cases} \tag{3.6}$$

• **Transmission delay:** the time it takes for the data collected at the g_i gateway to reach the sink [59]. Equation (3.7), the data transfer delay collected from gate g_i to sink D(g_i) is calculated. In this regard, d_q, d_p, and d_t show the queuing delay, propagation delay, and message transmission delay of the sending node of the intermediate links, respectively.

$$D(g_i) = (d_q + \mathrm{d}_t + d_p) \times \text{HopCount}(g_i) \tag{3.7}$$

According to the equation (3.7), it is directly related to the number of HopCount. For this purpose, MaxHop is considered in the cluster head merit function.

• **Maximum distance (MaxDist):** shows the maximum distance between neighboring nodes in the selected path [60]. The maximum distance is obtained using the equation (3.8).

$$\text{MaxDist} = \text{Max}\{\text{dis}(g_i,\text{NextHop}(g_i))|\forall \text{i},1 \le i \le \text{m}, \text{g}_i \in G\} \tag{3.8}$$

• **Maximum number of steps (MaxHop):** shows the maximum number of nodes in the selected path [61]. The maximum number of steps is obtained using the equation (3.9).

$$\text{MaxHop}=\text{Max}\{\text{HopCount}(g_i)|\forall \text{i},1 \le i \le \text{m}, \text{g}_i \in G\} \tag{3.9}$$

• $\boldsymbol{u_x}$ **and** $\boldsymbol{u_y}$: Respectively the highest limits of the location on the x and y coordinate axes. $\boldsymbol{l_x}$ and $\boldsymbol{l_y}$ are respectively the lowest limit of the location on the x and y coordinate axes.

3.3 Proposed cluster-based routing protocol by using MFO algorithm:

This protocol consists of two operational phases:

1- Clustering of sensors.

2- Routing of cluster head gateways.

In the proposed protocol, unlike the cluster-based routing protocol inspired by the meta-heuristic algorithm of particle swarm, first the clustering phase and then the routing phase are implemented. Protocol's primary objective in these two operational phases is to increase the longevity of the network by focus on sensor networks' power consumption problem. The protocol is inspired by the meta-heuristic Moth-Flame Algorithm, which is an enhancement technique that is based on the attitude of moths towards a light source. The Moth-Flame Algorithm has been shown to be efficient in finding optimal solutions in WSNs. The proposed protocol uses unbalanced clustering techniques to prevent the formation of energy holes, which can lead to the early death of nodes and data transfer issues. Unbalanced clustering involves calculating the cluster size depending on how far away each cluster is from the sink. If a cluster is located near to the sink, it will be smaller, and if it is farther away, it will be bigger and this helps to prevent the formation of energy holes.

3.3.1 Clustering phase:

In this phase, a clustering algorithm using MFO algorithm is presented. In most clustering algorithms, only the distance of the cluster head from the sink is considered, but in this research, in addition to the residual energy of the cluster head, the distance of the cluster members from the cluster head is also considered. Because one of the goals of the research is to increase the failure tolerance in the cluster heads. Figure (3.2) shows an example of clustering in this network model.

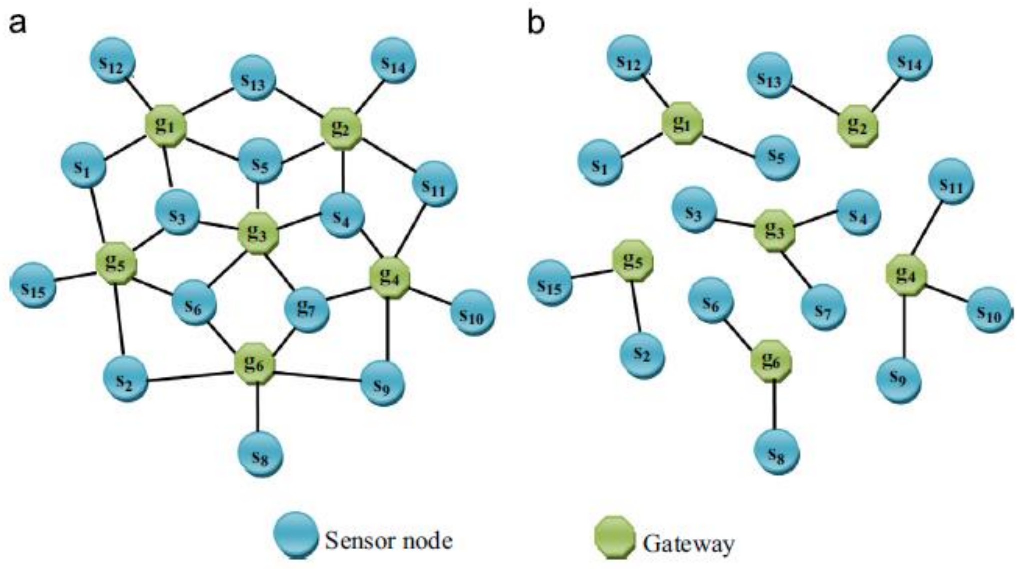

a
b
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} s_{15}
g_1 g_2 g_3 g_4 g_5 g_6 g_7
Sensor node
Gateway

Figure 3. 2 Wireless sensor network -a before clustering b- after clustering [3]

Equation (3.10) calculates the utility of selecting gateway g_j as the cluster head. Equation (3.11) also obtains the variable value of failure tolerance for gate g_j [11].

$$\text{Efficiency}_C(s_j,\text{M}_{\text{is}}) = \begin{cases} \frac{R(M_{\text{is}})\times E_{\text{residual}}(M_{\text{is}})}{d^2(s_j,\text{M}_{\text{is}})}, & \text{if } \text{d}(s_j, M_{\text{is}})<\text{d}_0 \\ \frac{R(M_{\text{is}})\times E_{\text{residual}}(M_{\text{is}})}{d^4(s_j,\text{M}_{\text{is}})}, & \text{if } \text{d}(s_j,\text{M}_{\text{is}}) \geq d_0 \end{cases} \tag{3.10}$$

$$R(M_{\text{is}}) = \text{N}_0 - \text{Members\_Number}(M_{\text{is}}) \tag{3.11}$$

The utility of the candidate answer is also equal to the sum of the utility of the sensors.

The proposed clustering algorithm is implemented in sink and consists of two steps:

1- Setup.

2- Optimizing candidate answers.

The initialization step (Setup) is executed only once, while the optimization step of the candidate solutions is executed until the equation (3.12) is satisfied. The fixed parameter **Max_clustering_iter** is the maximum execution stage in the clustering phase, which is considered in simulation 10.

$$\text{Clustering\_iter} \leq \text{Max\_clustering\_iter} \tag{3.12}$$

3.3.1.1 Setup step:

It puts Clustering_iter=0 at the beginning. Then, each gateway sensor S selects GoSS(s_i) from its transmission range as cluster head candidates. Each sensor s sends the identification numbers of the selected gateway along with its identification number to the sink in the form of a CM message. After a period of time, the sink examines the received CM (Cluster Message) messages from the sensors and obtains the candidate answers.

Then, an $M_{\text{Clustering\_iter}}$ matrix of order r*s (r is the number of candidate answers and s is the number of sensors) is built using the gates selected by the sensors so that $M_{\text{Clustering\_iter}}(i,s)$ is the identification number of the gate selected by the sensor s In the answer of candidate i in the Clustering_iter stage. r shows the size of the candidate answers and each row of the matrix M shows a candidate answer. Since Clustering_iter is zero, $M_{\text{Clustering\_iter}}$ is represented by M_0 in this step.

3.3.1.2 Optimizing candidate answer step:

The candidate solution optimization step is also done in sink. First, the Flame_No parameter is obtained using the equation (2.16). In this research, the maximum number of flames is considered to be N=5. Also, the maximum execution stage T=10 is considered.

$$\text{flame\_no} = \text{round}(\text{N-L} \times \frac{\text{N-1}}{T}) \tag{2.16}$$

If Clustering_iter=1:

- Sorts the elements of the utility vector of the $\text{OM}_{\text{Clustering\_iter}}$ matrix in descending order. The resulting vector $\text{OF}_{\text{Clusteirng\_iter}}$ is considered.
- The rows of the $\text{M}_{\text{Clustering\_iter}}$ matrix are also sorted according to the $\text{OF}_{\text{Clustering\_iter}}$ elements. The resulting matrix $\text{F}_{\text{Clustering\_iter}}$ is also considered. The $\text{F}_{\text{Clustering\_iter}}$ matrix and the $\text{OF}_{\text{Clustering\_iter}}$ vector are respectively the flame candidate answers and their utility vector in the Clustering_iter stage. The $\text{F\_X}_{\text{Clustering\_iter}}$ and $\text{F\_Y}_{\text{Clustering\_iter}}$ matrices are also the location matrices on the x and y coordinate axes in the Clustring_iter step.

If Clustering_iter > 1, do for each s-th variable from the i-th candidate answer:

- TM_X(i,s): Temporary matrix.
- If TM_X(i,s) is greater than u_x, we set TM_X(i,s) = u_x.
- If TM_X(i,s) is smaller than l_x, we set TM_X(i,s) = l_x.
- If TM_Y(i,s) is greater than u_y, we set TM_Y(i,s) = u_y.
- If TM_Y(i,s) is smaller than l_y, we set TM_Y(i,s) = l_y.
- By using the nearest gate technique and temporary locations TM_X(i,s) and TM_Y(i,s), Updates the locations of $\text{M\_X}_{\text{Clustering\_iter}}$(i,s) and $\text{M\_Y}_{\text{Clustering\_iter}}$(i,s).

TM_X (i,s) and TM_Y(i,s) are respectively the temporary locations of the gate selected by the s-th sensor in the i-th candidate answer on the x and y axes. In the closest gate technique, for each s-th variable of the i-th candidate solution, sink selects a gate from the GoSS(sj) set that has the smallest Euclidean distance from the location TM_X(i,s) and TM_Y(i,s). The equation (3.14) shows the Euclidean distance between locations.

$$\text{d}=\sqrt{(\text{M\_X}_{\text{Clustering\_iter}}(\text{i,s}) - \text{TM\_X}(\text{i,s}))^2 + (\text{M\_Y}_{\text{Clustering\_iter}}(\text{i,s}) - \text{TM\_Y}(\text{i,s}))^2} \tag{3.14}$$

Then, sink using equations (3.10), (3.11) and obtains the utility of each candidate solution M_{i} and constructs the $\text{OM}_{\text{Clustering\_iter}}$ vector.

If Clustering_iter>1:

- $OM_{Clustering\_iter}$ and $OM_{Clustering\_iter-1}$ vectors are combined and arranged in descending order.
- N elements from N_2 elements of $OM_{Clustering\_iter}$ and $OM_{Clustering\_iter-1}$ vectors are selected in order of usefulness and are considered as $OF_{Clustering\_iter}$ vector elements.
- Corresponding rows of $OF_{Clustering\_iter}$ vector are also found using $M_{Clustering\_iter}$ and $M_{Clustering\_iter-1}$ matrices and are considered as $F_{Clustering\_iter}$ matrix.

Next, the utility sink selects the first element of the $OF_{Clustering\_iter}$ vector as the optimal flame utility **Best_flame_fitness** and the first middle of the $F_{Clustering\_iter}$ matrix as the optimal flame vector **Best_flame_vector**. Then, for each s-th variable of each i-th answer, the candidate does the following:

If i <= flame_no:

- Obtains $DX_{Clustering\_iter}(i,s)$ and $DY_{Clustering\_iter}(i,s)$ using equations (3.15) and (3.16).

$$DX_{Clustering\_iter}(i,s) = |M\_X_{Clustering\_iter}(i,s) - F\_X_{Clustering\_iter}(i,s)| \quad (3.15)$$

$$DY_{Clustering\_iter}(i,s) = |M\_Y_{Clustering\_iter}(i,s) - F\_Y_{Clustering\_iter}(i,s)| \quad (3.16)$$

- b=1.
- Using equations (2.13) and (2.14), obtains the value of t.
- Using equations (3.17) and (3.18), TM_X(i,s) and TM_Y(i,s) are obtained.

$$TM\_X(i,s) = DX_{Clustering\_iter}(i,s).\, e^{bt}.\cos(2\pi t)+F\_X_{Clustering\_iter}(i,s) \quad (3.17)$$

$$TM\_Y(i,s) = DY_{Clustering\_iter}(i,s).\, e^{bt}.\cos(2\pi t)+F\_Y_{Clustering\_iter}(i,s) \quad (3.18)$$

If i > flame_no:

- Obtains $DX_{Clustering\_iter}(i,s)$ and $DY_{Clustering\_iter}(i,s)$ using equations (3.15) and (3.16).
- b=1.
- Using equations (2.13) and (2.14), obtains the value of t.
- Using equations (3.19) and (3.20), obtains TM_X(i, s) and TM_Y(i,s).

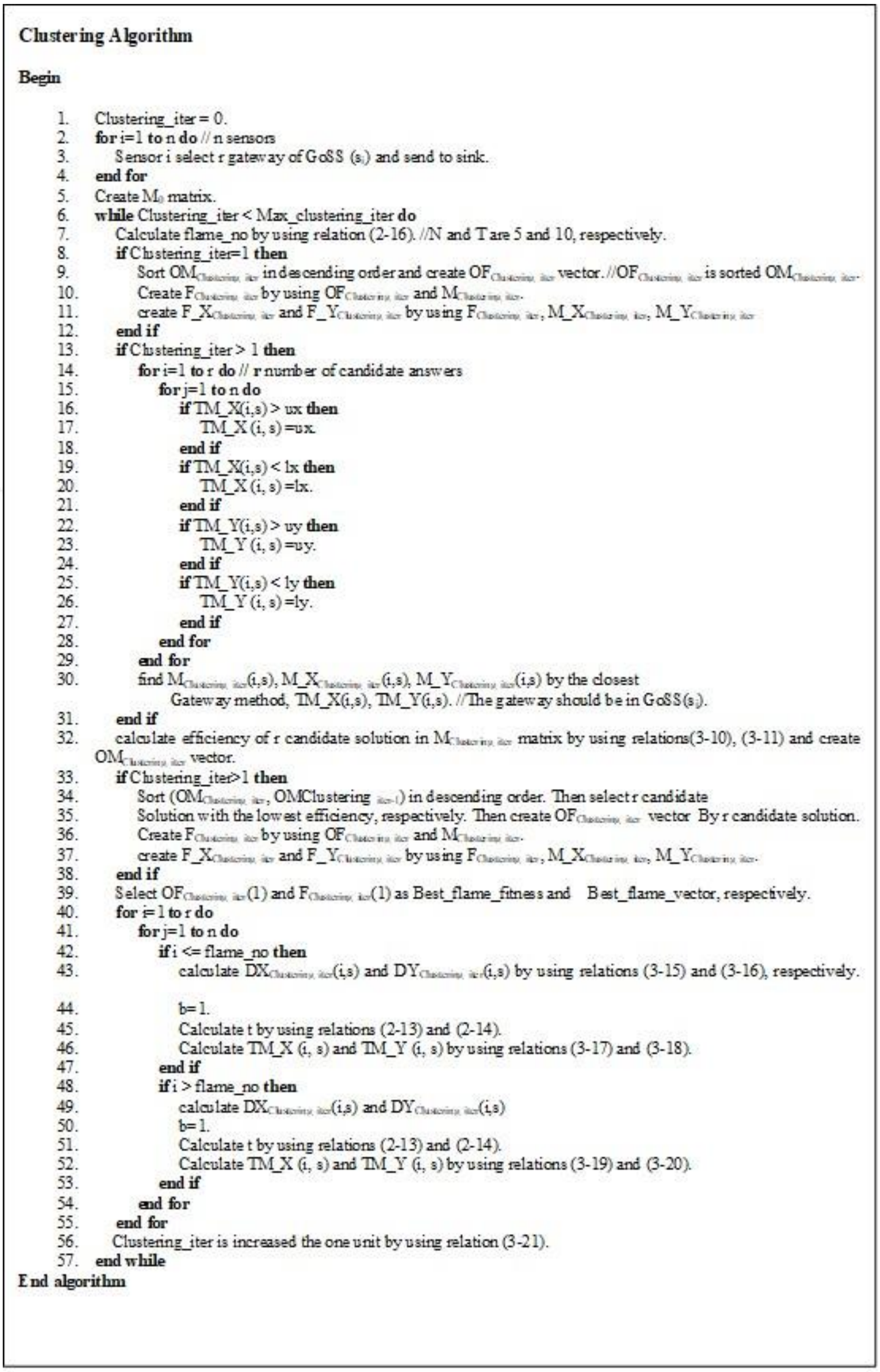
$$TM\_X(i,s) = DX_{Clustering\_iter}(i,s).\, e^{bt}.\cos(2\pi t)+F\_X_{Clustering\_iter}(flame\_no,s) \quad (3.19)$$

$$TM\_Y(i,s) = DY_{Clustering\_iter}(i,s).\, e^{bt}.\cos(2\pi t)+F\_Y_{Clustering\_iter}(flame\_no,s) \quad (3.20)$$

Then Clustering_iter is increased by one unit using equation (3.21).

$$\text{Clustering\_iter}=\text{Clustering\_iter}+1 \tag{3.21}$$

Next, the sink checks whether the equation (3.12) is valid. If the equation (3.12) is valid, the sink optimization phase is executed, otherwise, the clustering is finished and the best flame vector **Best_flame_vector** is selected as the best candidate solution. Figure (3.3) shows the clustering phase pseudo-code. In this pseudo-code, when the executor of the command is not specified, it means the sink.



Clustering Algorithm

Begin

```
1.  Clustering_iter = 0.
2.  for i=1 to n do // n sensors
3.      Sensor i select r gateway of GoSS ($s_i$) and send to sink.
4.  end for
5.  Create $M_0$ matrix.
6.  while Clustering_iter < Max_clustering_iter do
7.      Calculate flame_no by using relation (2-16). //N and T are 5 and 10, respectively.
8.      if Clustering_iter=1 then
9.          Sort $OM_{Clustering, iter}$ in descending order and create $OF_{Clustering, iter}$ vector. //$OF_{Clustering, iter}$ is sorted $OM_{Clustering, iter}$.
10.         Create $F_{Clustering, iter}$ by using $OF_{Clustering, iter}$ and $M_{Clustering, iter}$.
11.         create $F\_X_{Clustering, iter}$ and $F\_Y_{Clustering, iter}$ by using $F_{Clustering, iter}$, $M\_X_{Clustering, iter}$, $M\_Y_{Clustering, iter}$
12.     end if
13.     if Clustering_iter > 1 then
14.         for i=1 to r do // r number of candidate answers
15.             for j=1 to n do
16.                 if TM_X(i,s) > ux then
17.                     TM_X (i, s) =ux.
18.                 end if
19.                 if TM_X(i,s) < lx then
20.                     TM_X (i, s) =lx.
21.                 end if
22.                 if TM_Y(i,s) > uy then
23.                     TM_Y (i, s) =uy.
24.                 end if
25.                 if TM_Y(i,s) < ly then
26.                     TM_Y (i, s) =ly.
27.                 end if
28.             end for
29.         end for
30.         find $M_{Clustering, iter}$(i,s), $M\_X_{Clustering, iter}$(i,s), $M\_Y_{Clustering, iter}$(i,s) by the closest
                Gateway method, TM_X(i,s), TM_Y(i,s). //The gateway should be in GoSS($s_i$).
31.     end if
32.     calculate efficiency of r candidate solution in $M_{Clustering, iter}$ matrix by using relations(3-10), (3-11) and create
    $OM_{Clustering, iter}$ vector.
33.     if Clustering_iter>1 then
34.         Sort ($OM_{Clustering, iter}$, OMClustering $_{iter-1}$) in descending order. Then select r candidate
35.         Solution with the lowest efficiency, respectively. Then create $OF_{Clustering, iter}$ vector By r candidate solution.
36.         Create $F_{Clustering, iter}$ by using $OF_{Clustering, iter}$ and $M_{Clustering, iter}$.
37.         create $F\_X_{Clustering, iter}$ and $F\_Y_{Clustering, iter}$ by using $F_{Clustering, iter}$, $M\_X_{Clustering, iter}$, $M\_Y_{Clustering, iter}$.
38.     end if
39.     Select $OF_{Clustering, iter}$(1) and $F_{Clustering, iter}$(1) as Best_flame_fitness and  Best_flame_vector, respectively.
40.     for i= 1 to r do
41.         for j=1 to n do
42.             if i <= flame_no then
43.                 calculate $DX_{Clustering, iter}$(i,s) and $DY_{Clustering, iter}$(i,s) by using relations (3-15) and (3-16), respectively.

44.                 b=1.
45.                 Calculate t by using relations (2-13) and (2-14).
46.                 Calculate TM_X (i, s) and TM_Y (i, s) by using relations (3-17) and (3-18).
47.             end if
48.             if i > flame_no then
49.                 calculate $DX_{Clustering, iter}$(i,s) and $DY_{Clustering, iter}$(i,s)
50.                 b=1.
51.                 Calculate t by using relations (2-13) and (2-14).
52.                 Calculate TM_X (i, s) and TM_Y (i, s) by using relations (3-19) and (3-20).
53.             end if
54.         end for
55.     end for
56.     Clustering_iter is increased the one unit by using relation (3-21).
57. end while
```

End algorithm

Figure 3. 3 Cluster-based routing inspired by Moth-flame Optimization Algorithm

3.3.2 Routing phase:

For this phase, a routing algorithm inspired by the Moth-Flame optimization algorithm is presented. The routing algorithm, like the clustering algorithm, consists of two steps:

1- Setup.

2- Optimizing candidate answers.

In the routing algorithm, like the clustering algorithm, the setup step is executed only once, while the optimization step of the candidate solutions is executed until the equation (3.22) is established. The fixed parameter **Max_routing_iter** is the maximum execution phase in the routing phase.

$$\text{Routing\_iter} \leq \text{Max\_routing\_iter} \tag{3.22}$$

This routing algorithm, in addition to the merit function variables of the candidate answer, in the cluster-based routing protocol using the particle swarm optimization algorithm, the failure tolerance variable is also considered [11]. Equation (3.23) shows the merit function of candidate answer M_i.

$$\text{Efficiency}_R(M_i) = \frac{\text{Min\_R}}{(W_1 \times \text{Max\_Hop}) + (W_2 \times \text{Max\_Distance})} \tag{3.23}$$

In equation (3.23), Min_R shows the lowest failure tolerance of gate g_s in the i-th candidate solution. Equation (3.24) $R(M_{is})$ shows the failure tolerance of gate g_s in candidate solution M_i. Max_Distance shows the maximum Euclidean distance of the selected gates from the sink in the candidate answer Mi. Max_Hop also shows the maximum number of steps of the gates selected from the sink in the answer of candidate Mi. The fixed parameters W_1 and W_2 are coefficients of the routing merit function, which are considered to be between 0.8 and 0.2.

$$R(M_{\text{is}}) = B_0 - \text{NeighborGateway\_Count}(M_{\text{is}}) \tag{3.24}$$

In equation (3.24), parameter B_0 is the threshold limit of the number of gates that can choose a gate as the next gate to transmit data to the sink. NeighborGateway_Count (M_{is}) is the number of gateways that have selected M_{is} as the next gateway.

3.3.2.1 Setup step:

At first it sets Routing_iter=0. Then, each gate g_j selects the r number of gate candidate answers from its transmission range PNextHop (g_i) as candidates for the next gate. Each gateway g_j sends the identification numbers r of the chosen gateway along with its own identification number in the form of an NHGM (Next Hop Gateway Message) message to the sink. After a period of time, the sink examines the received NHGM messages from the gateways and obtains the candidate answers.

Then a matrix $M_{Routing\_iter}(i,j)$ of order r*m is constructed using the next gateways selected by the gateways such that $M_{Routing\_iter}(i,j)$ is the identification number of the next gateway selected by j-th gateway in i-th best candidate answer in the Routing_iter stage. r shows the size of the candidate answers and each row of the M matrix shows a candidate answer in the routing phase. Since Routing_iter is zero, $M_{Routing\_iter}$ is represented by M_0 in this step.

Both $M\_X_{Routing\_iter}$ and $M\_Y_{Routing\_iter}$ matrices are of r*m order and show the location of the next gates selected in the MRouting_iter matrix on the x and y coordinate axes. The registers $M\_X_{Routing\_iter}(i,j)$ and $M\_Y_{Routing\_iter}(i,j)$ are respectively the location of the gate selected as the next gate of the j-th gate in the i-th candidate answer on the x and y coordinate axes and in the Routing_iter step. The matrices $M\_X_{Routing\_iter}$ and $M\_Y_{Routing\_iter}$ are also represented by M_X_0 and M_Y_0 respectively.

3.3.2.2 Optimizing candidate answer step:

The step of optimizing candidate solution is also done in the sink. First, the **flame_no** parameter is obtained using the equation (2.16). In this research, the maximum number of flames is considered to be N=5. Also, the maximum execution stage T=10 is considered.

If Routing_iter=1:

- Sorts the elements of the $OM_{Routing\_iter}$ vector in descending order. The resulting vector $OF_{Routing\_iter}$ is considered.
- $M_{Routing\_iter}$ matrix rows are also sorted according to $OF_{Routing\_iter}$ elements. The resulting matrix $F_{Routing\_iter}$ is also considered. The $F_{Routing\_iter}$ matrix and the $OF_{Routing\_iter}$ vector are considered respectively as flame candidate solutions and

their utility vector in the Routing_iter stage. The $F\_X_{Routing\_iter}$ and $F\_Y_{Routing\_iter}$ matrices are also the location matrices on the x and y coordinate axes and in the Routing_iter stage.

If Routing_iter > 1, do for each j-th variable from the i-th candidate answer:

- If TM_X(i,j) is greater than ux, we set TM_X(i,j) = u_x.
- If TM_X(i,j) is smaller than lx, we set TM_X(i,j) = l_x.
- If TM_Y(i,j) is greater than uy, we set TM_Y(i,j) = u_y.
- If TM_Y(i,j) is smaller than ly, we set TM_Y(i,j) = l_y.
- Updates $M\_X_{Routing\_iter}(i,j)$ and $M\_Y_{Routing\_iter}(i,j)$ locations using the nearest gateway technique and temporary locations TM_X(i,j) and TM_Y(i,j).

TM_X(i,j) and TM_Y(i,j) are, respectively, the temporary locations of the next gate selected by the j-th gate in the i-th candidate answer on the x and y axes. In the technique of the nearest gate, sink for each j-th variable of the i-th candidate answer and using the equation (3.14), select a gate from the set PNextHop(gj) that has the smallest Euclidean distance from the location TM_X(i,j) and TM_Y(i,j).

Then sink obtains the utility of each candidate solution M_i by using equations (3.23) and (3.24) and constructs the $OM_{Routing\_iter}$ vector.

If Routing_iter>1:

- $OM_{Routing\_iter}$ and $OM_{Routing\_iter-1}$ vectors are combined and arranged in descending order.
- N elements from N_2 elements of $OM_{Routing\_iter}$ and $OM_{Routing\_iter-1}$ vectors are selected in order of usefulness and are considered as $OF_{Routing\_iter}$ vector elements.
- Corresponding rows of $OF_{Routing\_iter}$ vector are also found using $M_{Routing\_iter}$ and $M_{Routing\_iter-1}$ matrices and are considered as $F_{Routing\_iter}$ matrix.

Next, the utility sink selects the first element of the $OF_{Routing\_iter}$ vector as the optimal flame utility Best_flame_fitness and the first row of the $F_{Routing\_iter}$ matrix as the optimal flame vector Best_flame_vector. Then, for every j-th variable of every i-th answer, the candidate does the following:

If i<=flame_no:

- Obtains $DX_{Routing\_iter}(i,j)$ and $DY_{Routing\_iter}(i,j)$ using equations (3.25) and (3.26).

$$DX_{Routing\_iter}(i,j) = \left(\sum\left(M\_X_{Routing\_iter}(i,j) - F\_X_{Routing\_iter}(i,j)\right)^2\right)^{\frac{1}{2}} \qquad (3.25)$$

$$DY_{Routing\_iter}(i,j) = \left(\sum\left(M\_Y_{Routing\_iter}(i,j) - F\_Y_{Routing\_iter}(i,j)\right)^2\right)^{\frac{1}{2}} \qquad (3.26)$$

- b=1.
- Using equations (2.13) and (2.14), obtains the value of t.
- Using equations (3.27) and (3.28), obtains TM_X(i,j) and TM_Y(i,j).

$$TM\_X(i,j) = DX_{Routing\_iter}(i,j).\,e^{bt}.\cos(2\pi t)+F\_X_{Routing\_iter}(i,j) \qquad (3.27)$$

$$TM\_Y(i,j) = DY_{Routing\_iter}(i,j).\,e^{bt}.\cos(2\pi t)+F\_Y_{Routing\_iter}(i,j) \qquad (3.28)$$

If i > flame_no:

- Obtains $DX_{Routing\_iter}(i,j)$ and $DY_{Routing\_iter}(i,j)$ using equations (3.25) and (3.26).
- b=1.
- Using equations (2.13) and (2.14), obtains the value of t.
- Using equations (3.29) and (3.30), obtains TM_X(i,j) and TM_Y(i,j).

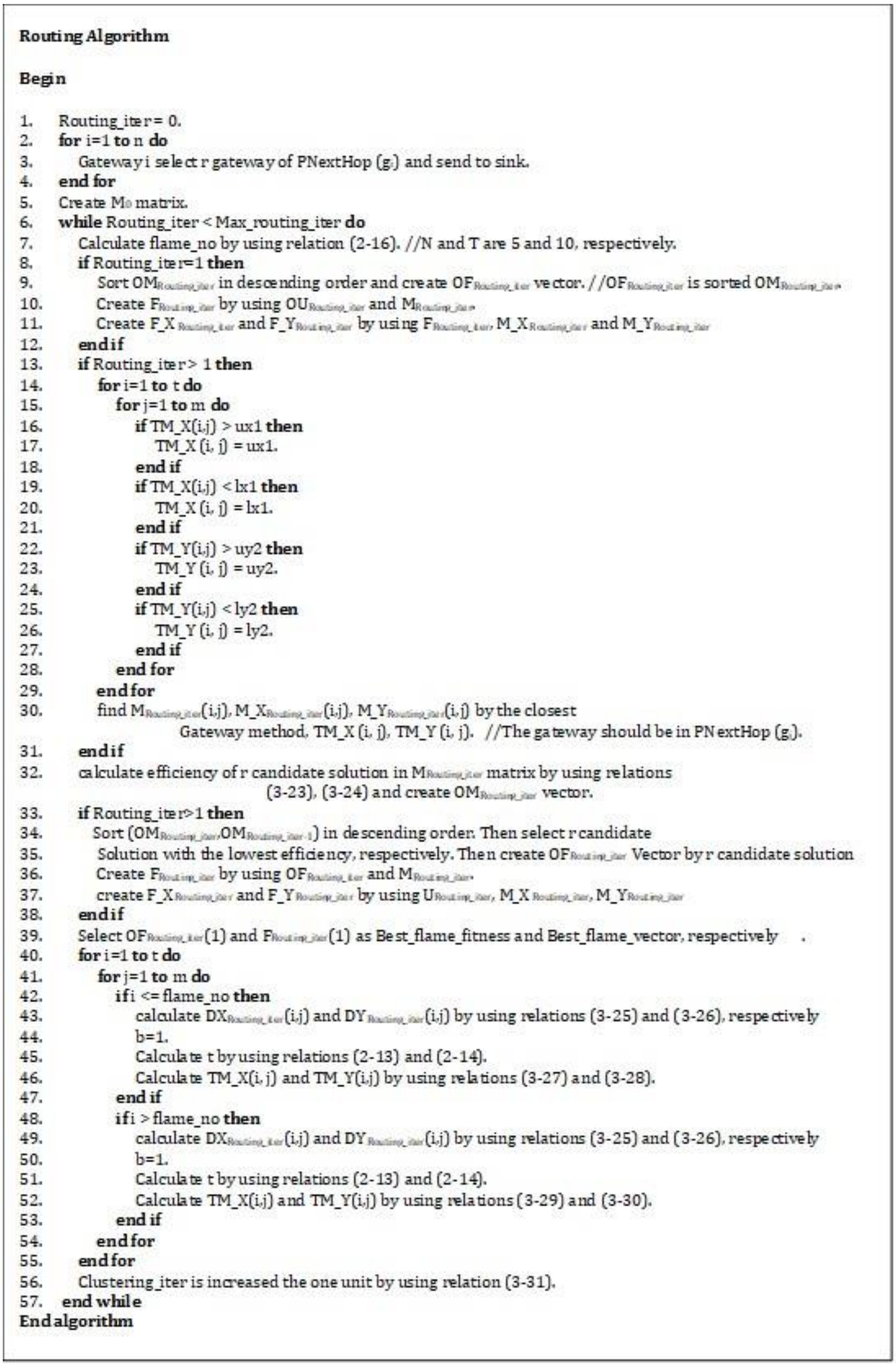
$$TM\_X(i,j) = DX_{Routing\_iter}(i,j).\,e^{bt}.\cos(2\pi t)+F\_X_{Routing\_iter}(flame\_no,j) \qquad (3.29)$$

$$TM\_Y(i,j) = DY_{Routing\_iter}(i,j).\,e^{bt}.\cos(2\pi t)+F\_Y_{Routing\_iter}(flame\_no,j) \qquad (3.30)$$

Then it is increased by one unit using the equation (3.31).

$$Routing\_iter=Routing\_iter + 1 \qquad (3.31)$$

Next, sink checks if the equation (3.22) holds. If the equation (3.22) hold, the sink optimization phase is executed. Otherwise, the routing is finished and the best flame vector **Best_flame_vector** is selected as the best candidate solution. Figure (3.4) shows the routing phase pseudo-code. In this pseudo-code, when the executor of the command is not specified, it means the sink.



```
Routing Algorithm

Begin

1.  Routing_iter= 0.
2.  for i=1 to n do
3.     Gateway i select r gateway of PNextHop ($g_i$) and send to sink.
4.  end for
5.  Create $M_0$ matrix.
6.  while Routing_iter < Max_routing_iter do
7.     Calculate flame_no by using relation (2-16). //N and T are 5 and 10, respectively.
8.     if Routing_iter=1 then
9.        Sort $OM_{Routing\_iter}$ in descending order and create $OF_{Routing\_iter}$ vector. //$OF_{Routing\_iter}$ is sorted $OM_{Routing\_iter}$.
10.       Create $F_{Routing\_iter}$ by using $OU_{Routing\_iter}$ and $M_{Routing\_iter}$.
11.       Create $F\_X_{Routing\_iter}$ and $F\_Y_{Routing\_iter}$ by using $F_{Routing\_iter}$, $M\_X_{Routing\_iter}$ and $M\_Y_{Routing\_iter}$
12.    endif
13.    if Routing_iter> 1 then
14.       for i=1 to t do
15.          for j=1 to m do
16.             if TM_X(i,j) > ux1 then
17.                TM_X (i, j) = ux1.
18.             end if
19.             if TM_X(i,j) < lx1 then
20.                TM_X (i, j) = lx1.
21.             end if
22.             if TM_Y(i,j) > uy2 then
23.                TM_Y (i, j) = uy2.
24.             end if
25.             if TM_Y(i,j) < ly2 then
26.                TM_Y (i, j) = ly2.
27.             end if
28.          end for
29.       end for
30.       find $M_{Routing\_iter}$(i,j), $M\_X_{Routing\_iter}$(i,j), $M\_Y_{Routing\_iter}$(i,j) by the closest
                   Gateway method, TM_X (i, j), TM_Y (i, j).  //The gateway should be in PNextHop ($g_i$).
31.    endif
32.    calculate efficiency of r candidate solution in $M_{Routing\_iter}$ matrix by using relations
                          (3-23), (3-24) and create $OM_{Routing\_iter}$ vector.
33.    if Routing_iter>1 then
34.       Sort ($OM_{Routing\_iter}$,$OM_{Routing\_iter-1}$) in descending order. Then select r candidate
35.       Solution with the lowest efficiency, respectively. Then create $OF_{Routing\_iter}$ Vector by r candidate solution
36.       Create $F_{Routing\_iter}$ by using $OF_{Routing\_iter}$ and $M_{Routing\_iter}$.
37.       create $F\_X_{Routing\_iter}$ and $F\_Y_{Routing\_iter}$ by using $U_{Routing\_iter}$, $M\_X_{Routing\_iter}$, $M\_Y_{Routing\_iter}$
38.    endif
39.    Select $OF_{Routing\_iter}$(1) and $F_{Routing\_iter}$(1) as Best_flame_fitness and Best_flame_vector, respectively   .
40.    for i=1 to t do
41.       for j=1 to m do
42.          if i <= flame_no then
43.             calculate $DX_{Routing\_iter}$(i,j) and $DY_{Routing\_iter}$(i,j) by using relations (3-25) and (3-26), respectively
44.             b=1.
45.             Calculate t by using relations (2-13) and (2-14).
46.             Calculate TM_X(i, j) and TM_Y(i,j) by using relations (3-27) and (3-28).
47.          end if
48.          if i > flame_no then
49.             calculate $DX_{Routing\_iter}$(i,j) and $DY_{Routing\_iter}$(i,j) by using relations (3-25) and (3-26), respectively
50.             b=1.
51.             Calculate t by using relations (2-13) and (2-14).
52.             Calculate TM_X(i,j) and TM_Y(i,j) by using relations (3-29) and (3-30).
53.          end if
54.       end for
55.    end for
56.    Clustering_iter is increased the one unit by using relation (3-31).
57. end while
End algorithm
```

Figure 3. 4 Cluster-based routing inspired by Moth-flame optimization algorithm

3.4 Conclusion:

In this chapter, a cluster-based routing protocol is presented and the proposed routing protocol consists of two operational phases: 1- Clustering 2- Routing. Moth-Flame optimization algorithm has been used in both phases. The evaluation shows that the meta-heuristic Moth-Flame optimization (MFO) algorithm has better results in many cases than the meta-heuristic algorithms of particle swarm (PSO), gravitational search algorithm (GSA), bat algorithm (BA), flower pollination algorithm (FBA), States of Matter Search (SMS), firefly algorithm (FA), and genetics algorithm (GA), has earned. The structure of the algorithm is the same in both phases and it is designed based on the Moth-Flame optimization algorithm. However, in the clustering phase, the candidate solution elements are the cluster heads selected by the sensors, while in the routing phase, the candidate solution elements are the selected gateways as the next step in the path of data transmission from the gate to the sink. The implementation steps of the Moth-Flame optimization algorithm are the same in both clustering and routing phases. In the clustering and routing phases, the gateway closest to the temporary location is selected as the cluster head and the next gateway to send data to the sink, respectively. In the clustering phase, the shorter the distance between the cluster head and the sink, the lower the transmission delay and energy consumption. Likewise, in the routing phase, the shorter the distance between the next step gate and the sink, the shorter the path for data transmission and the lower the delay and energy consumption required for data transmission.

4
RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the proposed protocol based on (MFO) and the cluster-based routing protocol based on (PSO) [3] are evaluated. In the (MFO) routing protocol in addition to the distance of the cluster head from the sink and the distance of the cluster members from the cluster heads is also considered and this protocol is presented with the aim of reducing energy consumption and increasing network life, also in this protocol in addition to wireless sensors, a number of nodes around the sink are assumed and these nodes are called relay gateways, these nodes are responsible for receiving and sending the data received from the sensors to the sink, in addition to increasing the lifetime, this protocol also prevents redundancy in data transmission by using clustering. In this protocol, each sensor sends information received from the environment only to its cluster head gateway, the cluster head gateway also sends the information received from the sensors to the sink using intermediate gateways, although in this (MFO) algorithm has significant efficiency compared to (PSO) algorithm in terms of network lifetime and energy consumption.

4.2 Evaluation Method

To evaluate the proposed protocol (MFO) and compare it with the (PSO) protocol, we use OMNET++ simulation software version 4.5 to check the results and two scenarios are considered to analyze the results. The residual energy of the node, the distance of the cluster head node from the sink, the distance of the cluster member nodes from the cluster head and the failure tolerance parameter are considered as evaluation parameters. The evaluation criteria are average of Transmission delay, packet delivery rate, standard deviation of energy consumption, lifetime of gateways and sensors.

4.2.1 Evaluation Parameters

Node residual energy: Each sensor has an initial energy stored in its battery, every time a sensor sends or receives data, some of this energy is consumed, the residual energy shows the current amount of sensor energy, which is the initial energy minus the energy used for data transmission so far [62].

The distance of the cluster head node from the sink: shows the Euclidean distance between the cluster head node and the sink, assuming (X_S, Y_S) and (X_{CH}, Y_{CH}) as the Euclidean coordinates of the sink and cluster head, respectively, the Euclidean distance between them is calculated using the equation (4.1) [63].

$$d(\text{CH,S}) = \sqrt{(x_S - x_{\text{CH}})^2 + (y_S - y_{\text{CH}})^2} \tag{4.1}$$

The distance of the cluster member node from the cluster head node: shows the Euclidean distance between the cluster member and the cluster head gate, assuming (X_{Mem}, Y_{Mem}) and (X_{CH}, Y_{CH}) as the Euclidean coordinates of the cluster member and cluster head, respectively, the Euclidean distance between them is calculated using equation (4.2) [63].

$$d(\text{Mem,CH}) = \sqrt{(x_{\text{CH}} - x_{\text{Mem}})^2 + (y_{\text{CH}} - y_{\text{Mem}})^2} \tag{4.2}$$

Failure tolerance: the ability of a system to continue functioning and providing service in the event of one or more component failures. It is often measured by the number of failed components that a system can tolerate before losing overall functionality, it can also refer to the difference between the number of cluster members and the threshold value, where a threshold value is a predefined limit of allowable failures before the system no longer functions [64].

4.2.2 Evaluation Criteria:

Transmission delay: refers to the time it takes for a signal to be transmitted from the source to the destination, it can be affected by factors such as distance, interference, and network congestion. Reducing transmission delay is important for applications that require low latency, such as real-time monitoring and control. Techniques to reduce transmission delay include using shorter packet sizes, reducing the number of network hops, and implementing efficient routing algorithms [65]. In this research, the duration of data transfer from the sensor node to the sink is considered as delay.

Packet Delivery Rate: shows the number of received messages compared to the number of sent messages, The higher the number of lost messages, the lower the data delivery rate [66].

Standard deviation of energy consumption: It shows the energy consumption standard deviation for data transmission on the sensors and gateways. The greater the distance between the sensors and the sink, or the greater the distance between the gateways and the sink, the energy consumption increases [67]. However, we should not always consider the shortest distance, because there are cases where the shortest path from a number of sensors to the sink is a common path and this will increase data transmission congestion and increase data transmission queue latency, as a result, the data transfer delay increases.

Gateway lifetime: In this research, the network lifetime is considered as the ratio of remaining energy to energy consumption, equation (4.3) shows the lifetime of the network and in this regard, Consumption Energy shows the amount of recent energy consumption in the ith gateway and residual Energy also shows the remaining energy of the ith gate before the last energy consumption [68].

$$L(i) = \frac{\text{residual Energy}}{\text{Consumption Energy}} \tag{4.3}$$

Sensor lifetime: The time it takes for the first sensor to die, each sensor sends information received from its surroundings to the sink, if a sensor dies, information about its surroundings is no longer accessible and for this reason, network lifetime is considered as one of the most important criteria for evaluating routing and clustering protocols in wireless sensor networks [69].

4.2.3 The simulation environment and its settings:

The scenarios used in the research were based on a study that used the particle swarm optimization algorithm for cluster-based routing protocols [3]. The sensors and gateways were randomly distributed in a two-dimensional space of 1050x1050, with the gateways placed in a mesh topology, and the sensors are randomly distributed on two-dimensional space (X, Y), Specifically, it sets the minimum and maximum values of the X and Y coordinates of the two-dimensional space in which the sensors move. The values used in the code set the minimum and maximum X coordinates to 1m and 999m, respectively, and the minimum and maximum Y coordinates to 1m and 999m, respectively. These constraints ensure that the sensors do not move outside the specified area during the simulation. The initial energy levels for sensors and gateways were set at 2J and 10J, respectively, and the data message size was 24 bytes. The simulation time was set to 150 seconds, and two evaluation scenarios were used: the first evaluated the protocols' efficiency with 200, 300, 400, and 500 sensors and 64 gateways, while the second evaluated the efficiency with 64, 81, and 100 gateways and 500 sensors. In both scenarios, the sink was located at the coordinates (1020M, 1020M) outside the network. The OMNET++ version 4.5 was used to evaluate the proposed MFO protocol.

4.3 Evaluation Results

In this section, the proposed routing protocols (MFO) and particle swarm cluster-based routing are implemented in two scenarios and evaluation criteria are examined.

4.3.1 Evaluation of the proposed protocol (MFO) based on the first scenario:

In the first scenario, the evaluation criteria have been investigated on 4 wireless sensor networks with the number of sensors 200, 300, 400 and 500. In this scenario, it is assumed that the number of gates is 64 and it is also assumed that the sink is located outside the network at the location (1020M,1020M). The evaluation criteria are average of transmission delay, packet delivery rate, standard deviation of energy consumption, lifetime of gateways and sensors.

4.3.1.1 Evaluation and comparison of the proposed protocol (MFO) based on the average transmission delay in the first scenario:

In figure (4.1), the average transmission delay of the proposed routing protocol (MFO) and the routing protocol based on the particle swarm (PSO) in the first scenario are shown below. The horizontal axis represents the number of sensors in the network, which are 200, 300, 400 and 500 respectively and the vertical axis shows the average data transmission delay for this number of network sensors.

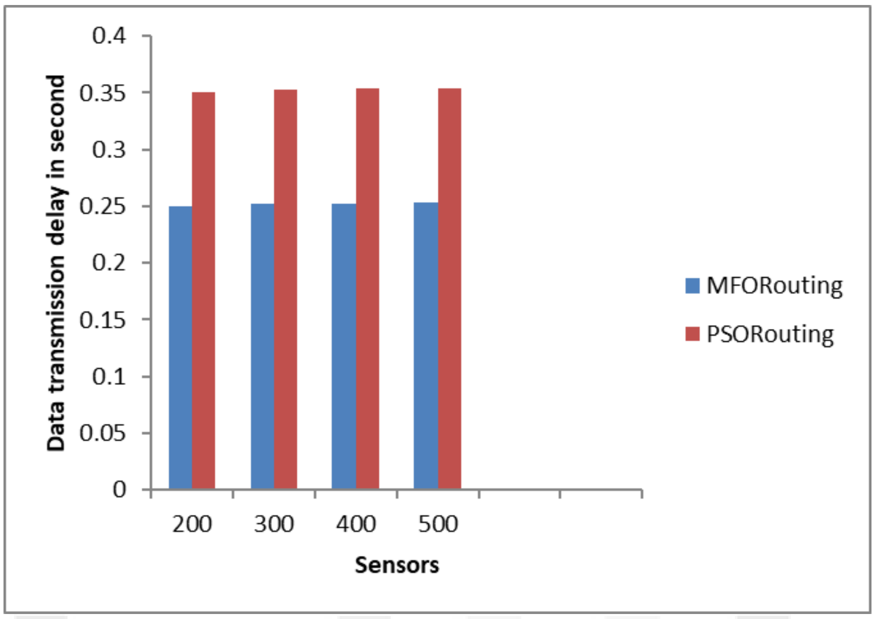


Figure 4. 1 Data transmission delay of the MFO and PSO routing protocols

Table 4. 1 Simulation results based on Data transmission delay

	PSO-Routing	MFO-Routing
200	0.350241	0.249694
300	0.352245	0.251679
400	0.353247	0.252578
500	0.354081	0.253026

In the clustering and routing phases, the proposed routing protocol (MFO) has a parameter called failure tolerance, this parameter makes the number of members of the cluster heads as well as the number of gateways that have chosen a gateway as the next gateway not to exceed a threshold if possible. In the clustering phase, some sensors are forced to choose a nearest gateway as the cluster head, therefore, a gateway may choose a gateway closer to the sink as the next gateway and this decreases the transmission delay. Because the proposed routing protocol (MFO) pays more attention to the energy consumption of gateways and sensors to increase the lifetime, it has selected closer nodes in some cases. As a result, the number of gateway nodes from cluster head to sink has decreased and this decreases the transmission delay. The simulation results show that the proposed routing protocol (MFO) has decreased the average transmission delay by 41% compared to the (PSO) protocol.

4.3.1.2 Evaluation and comparison of the proposed protocol (MFO) based on the packet delivery rate in the first scenario:

In figure (4.2), the packet delivery rate of the (MFO) and the routing protocol based on (PSO) in the first scenario are shown below, the horizontal axis represents the number of sensors in the network and the vertical axis shows the packet delivery rate for this number of network sensors.

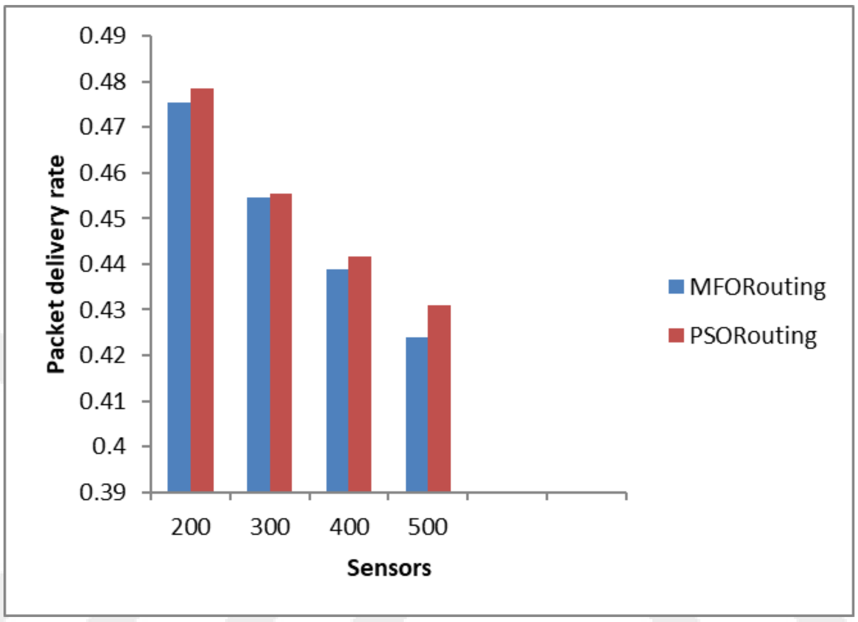


Figure 4. 2 Packet delivery rate of the MFO and PSO routing protocols

Table 4. 2 Simulation results based on Packet delivery rate

	PSO-Routing	MFO-Routing
200	0.478519	0.475519
300	0.455518	0.454518
400	0.441548	0.438846
500	0.430955	0.423943

As mentioned in the average data transmission delay, as long as all gateways are alive in the particle swarm cluster routing protocol (PSO), the message delivery rate in the particle swarm cluster routing protocol (PSO) is higher than Moth-flame cluster routing protocol (MFO). On the other hand, the proposed routing protocol (MFO) has a higher lifetime than (PSO) routing protocol, therefore, the intermediate gateways in this routing protocol

(MFO) survive longer than the routing protocol based on the (PSO), Therefore, when a number of gateways die in the particle swarm cluster-based routing protocol, some sensors can no longer send their data messages to the sink. The same simulation results show that the routing protocol based on the Moth-flame cluster has reduced the data delivery rate by one percent compared to the routing protocol based on the particle swarm cluster.

4.3.1.3 Evaluation and comparison of the proposed protocol (MFO) based on the standard deviation of energy consumption of the gateways in the first scenario:

In figure (4.3), the standard deviation of the energy consumption of the proposed protocol (MFO) and the routing protocol based on the (PSO) on the network gateways in the first scenario are shown below. The horizontal axis represents the number of sensors in the network and the vertical axis shows the standard deviation of energy consumption of network gateways.

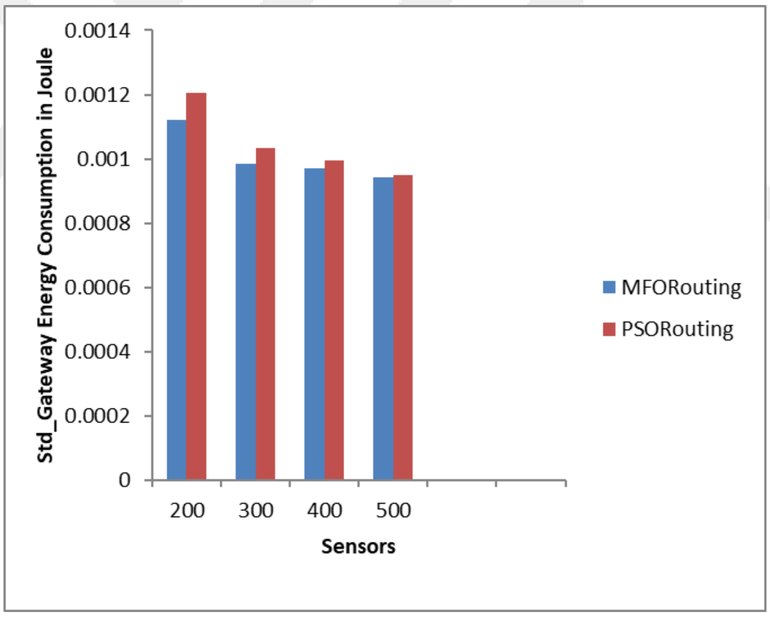


Figure 4. 3 Standard deviation of the gateway energy consumption of the MFO and PSO routing protocols

Table 4. 3 Simulation results based on standard deviation of the gateway energy consumption

	PSO-Routing	MFO-Routing
200	0.001205	0.001122
300	0.001032	0.000983
400	0.000994	0.000972
500	0.000951	0.000941

As shown in figure (4.3) and table (4.3), the routing protocol based on the Moth-flame compared to the routing protocol based on the particle swarm has been able to reduce the standard deviation of energy consumption by nine percent, because the proposed routing protocol (MFO) has created a balance in the energy consumption of gateways by applying the failure tolerance parameter.

4.3.1.4 Evaluation and comparison of the proposed protocol (MFO) based on the network lifetime in the first scenario:

Figure (4.4) shows the lifetime of the proposed protocol (MFO) and the routing protocol based on the (PSO) on the network gateways in the first scenario. The horizontal axis represents the number of sensors in the network and the vertical axis represents the lifetime of the network.

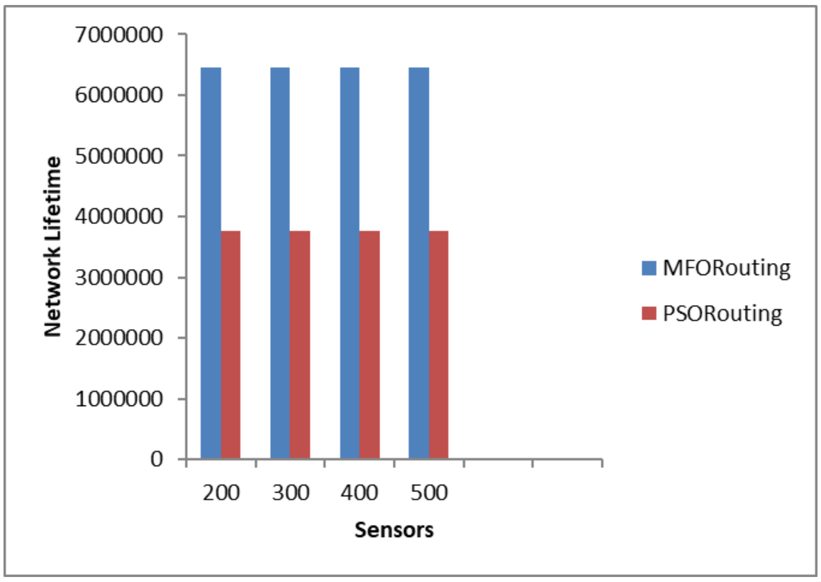


Figure 4. 4 Network Lifetime of the MFO and PSO routing protocols

Table 4. 4 Simulation results based on Network Lifetime

	PSO-Routing	MFO-Routing
200	3766716	6455461
300	3756715	6455460
400	3756712	6455430
500	3756710	6455421

As shown in figure (4.4), the lifetime of the gateways in the proposed routing protocol (MFO) is increased by 52% compared to the particle swarm routing protocol (PSO). From the data provided in table (4.4), it appears that the MFO-Routing protocol has a longer network lifetime compared to the PSO-Routing protocol. As the number of sensors in the network increases from 200 to 500, the lifetime of the network using MFO-Routing remains relatively consistent at around 6,455,461 while the lifetime of the network using PSO-Routing decreases from 3766716 to 3756710. It seems that, MFO-Routing is more suitable to use under more number of sensors.

4.3.2 Evaluation of the proposed protocol (MFO) based on the second scenario:

In the second scenario, the evaluation criteria have been investigated on three wireless sensor networks with the number of gateways 64, 81 and 100. In this scenario, it is assumed that the number of sensors is 500 and it is also assumed that the sink is located outside the network at the location (1020, 1020). In the following order, average of transmission delay, packet delivery rate, standard deviation of energy consumption, lifetime of gateways and sensors.

4.3.2.1 Evaluation and comparison of the proposed protocol (MFO) based on the average transmission delay in the second scenario:

In figure (4.5), the average transmission delay of the (MFO) and (PSO) in the second scenario is shown below. The horizontal axis shows the number of gateways in the network and the vertical axis also shows the average transmission delay on 64, 81 and 100 gateways in seconds, according to what was said for the average transmission delay in the second scenario, the transmission delay of the routing protocol based on the Moth-flame is lower than the routing protocol based on the particle swarm. However, since the number of gates increases from 64 to 100 in this scenario, the number of gateways near the sink increases. When the number of gateways increases, the number of paths increases with less distance and number of steps. This issue makes shorter routes with a smaller number of steps to be selected in the routing phase. Therefore, as the number of gateways increases, the average transmission delay difference between the two routing protocols based on (MFO) and (PSO) decreases.

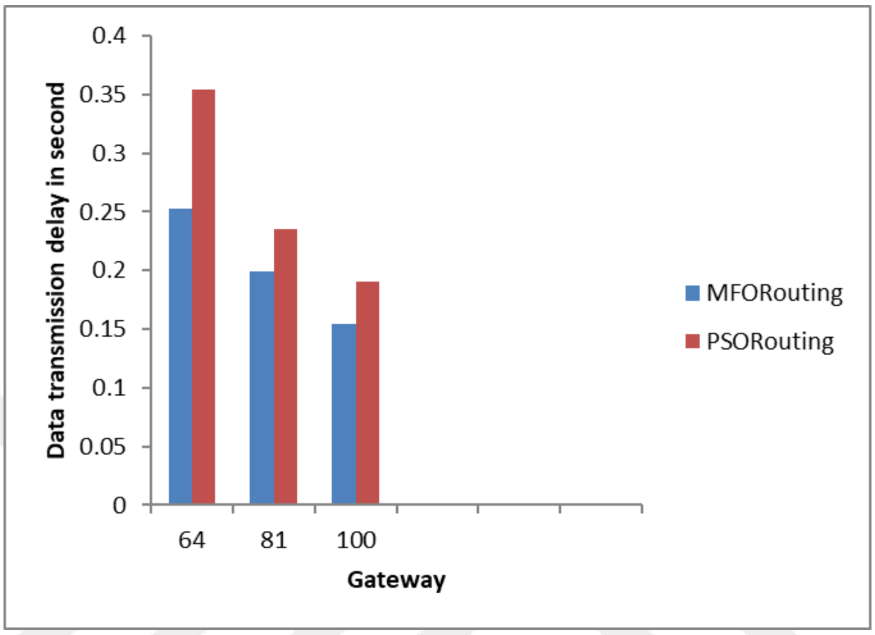


Figure 4. 5 Data transmission delay of the MFO and PSO routing protocols in second scenario

Table 4. 5 Simulation results based on Data transmission delay in second scenario

	PSO-Routing	MFO-Routing
64	0.354093	0.253032
81	0.234698	0.198734
100	0.190384	0.153949

The simulation results show that the average transmission delay in the routing protocol based on the (MFO) has decreased by 58% compared to the routing protocol based on the (PSO).

4.3.2.2 Evaluation and comparison of the proposed protocol (MFO) based on the packet delivery rate in the second scenario:

In figure (4.6), the packet delivery rate of the (MFO) and (PSO) are shown below in the second scenario. The horizontal axis represents the number of gateways in the network and the vertical axis represents the packet delivery rate. As shown in the figure, as the number of gateways increases, the number of routes with less distance and step count increases. Therefore, as the number of gateways increases, the average packet delivery rate difference between the (MFO) and (PSO) is reduced.

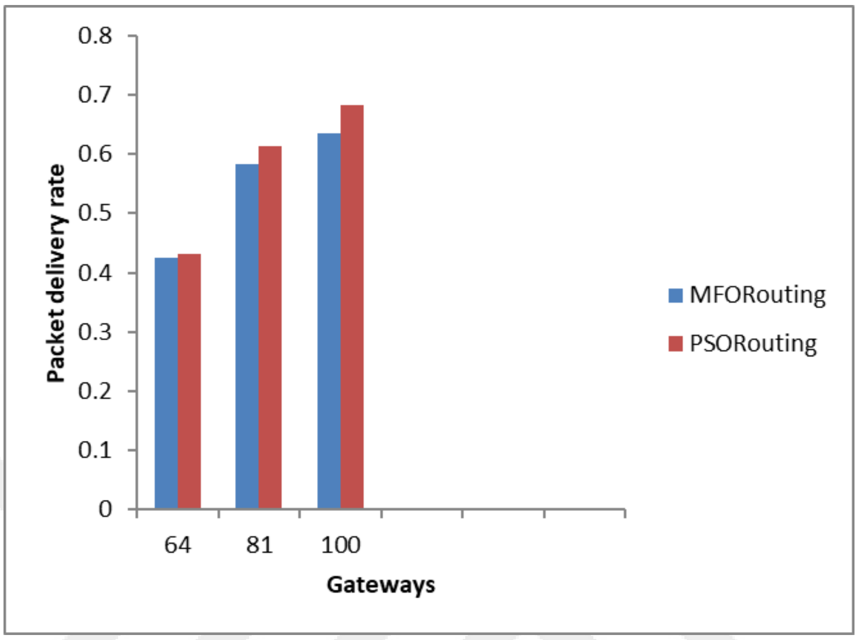


Figure 4. 6 Packet delivery rate of the MFO and PSO routing protocols in second scenario

Table 4. 6 Simulation results based on Packet delivery rate in second scenario

	PSO-Routing	MFO-Routing
64	0.430959	0.423951
81	0.612461	0.583512
100	0.682041	0.633863

As shown in the results, the routing protocol based on the (MFO) has reduced the data delivery rate by 17% compared to the routing protocol based on the (PSO).

4.3.2.3 Evaluation and comparison of the proposed protocol (MFO) based on the standard deviation of energy consumption of the gateways in the second scenario:

In figure (4.7), the standard deviation of the energy consumption of the (MFO) and (PSO) is shown below. The horizontal axis represents the number of gateways in the network and the vertical axis represents the standard deviation of energy consumption of the gateways. As the figure shows the number of gateways increases, the number of paths increases with the distance and number of steps less. Therefore, as the number of gateways increases, the standard deviation of energy consumption of the gateways is reduced between two routing protocols based on (MFO) and (PSO).

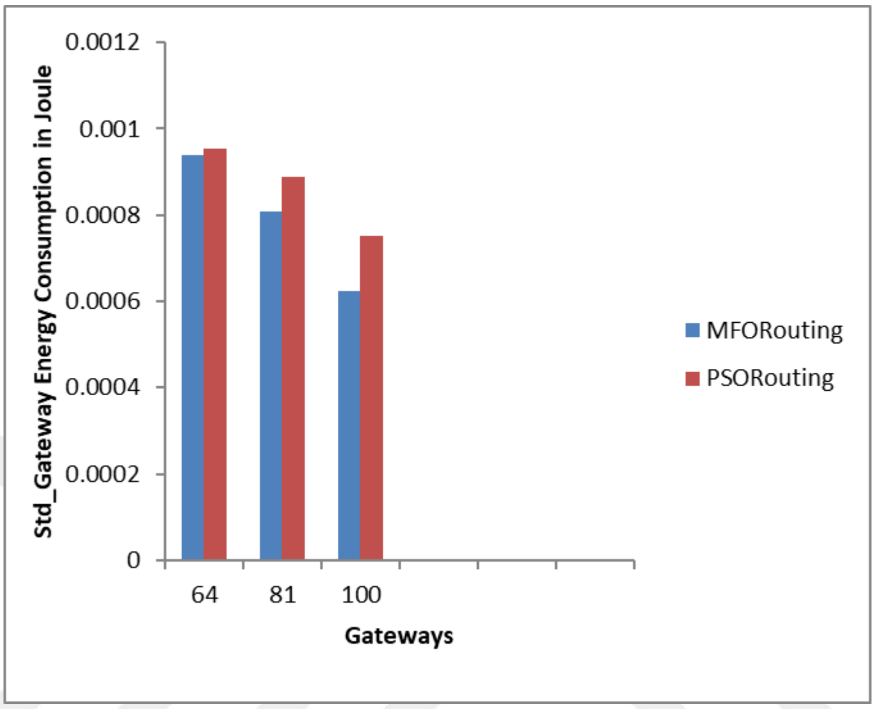


Figure 4. 7 Standard deviation of the gateway energy consumption of the MFO and PSO routing protocols in second scenario

Table 4. 7 Simulation results based on standard deviation of the gateway energy consumption in second scenario

	PSO-Routing	MFO-Routing
64	0.000953	0.000939
81	0.000887	0.000807
100	0.000752	0.000623

As shown in figure (4.7) and table (4.7), the routing protocol based on the (MFO) compared to the routing protocol based on the (PSO) in the second scenario was able to reduce the energy consumption standard deviation by almost 2%.

4.3.2.4 Evaluation and comparison of the proposed protocol (MFO) based on the network lifetime in the second scenario:

In figure (4.8) shows the network lifetime of the (MFO) and (PSO) on the network gateways in the second scenario. The horizontal axis represents the number of gateways in the network and the vertical axis represents the lifetime of the network gateways. However, when the number of gateways increases in this scenario, the difference between the lifetimes of these two protocols decreases because the failure tolerance parameter has a higher efficiency when the number of gates is less.

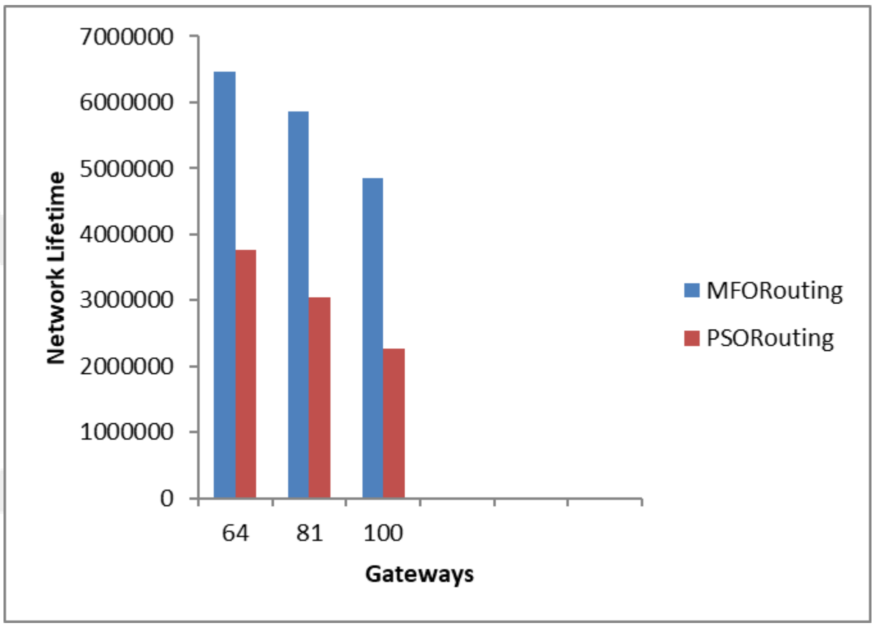


Figure 4. 8 Network Lifetime of the MFO and PSO routing protocols in second scenario

Table 4. 8 Simulation results based on Network Lifetime in second scenario

	PSO-Routing	MFO-Routing
64	3756710	6455451
81	4723613	5855460
100	4246715	4855461

As shown in figure (4.8) and table (4.8), the lifetime of the gateways in the (MFO) is increased by 34% compared to the (PSO) routing protocol.

4.4 Conclusion

The purpose of this research is to increase the lifetime of the network and reduce the delay in sending data from the sensors to the sink and for this reason, the routing protocols based on the (MFO) were investigated first. As mentioned in the second chapter, most of the routing and clustering protocols considered the remaining energy of the cluster head and the distance of the cluster head from the base station (Sink) and did not consider the remaining energy of the cluster members and the distance of the cluster members from the cluster head.

The other thing is to use Meta-Heuristic algorithms such as genetics, ants, particle swarms, etc. And the total efficiency of Meta-Heuristic algorithms is the same when applied to all optimization problems, therefore, a Meta-Heuristic algorithm may have high efficiency in one optimization problem and low efficiency in another optimization problem and this issue necessitates the design of new Meta-Heuristic protocols.

According to what was said, in this research, we have tried to increase the lifetime of the network and reduce the data transmission delay by taking into account the above three issues and for this reason, we have used Meta-Heuristic Moth-Flame algorithm.

The study aimed to evaluate the performance of two meta-heuristic algorithms, the Moth-Flame algorithm and the Particle-Swarm algorithm, in wireless sensor networks. The simulation results showed that the Moth-Flame algorithm exhibited better performance in a majority of cases when compared to the Particle-Swarm algorithm and this is because the Moth-Flame algorithm incorporates the characteristics of the moth's behavior in its search for the optimal solution, which leads to a more efficient search process and better results.

The simulation shows that the Meta-Heuristic Moth-Flame algorithm has better results in many cases than the Meta-Heuristic Particle-Swarm algorithm. The evaluation results show that the (MFO) routing protocol has increased network lifetime and equity in energy consumption on gateways. On the other hand, because in the (MFO) routing protocol, the energy consumption of the gateways and the failure tolerance parameter are seriously considered, in some cases the sensors in the clustering phase and the gateways in the routing phase choose the node that is closer. As a result, the distance or the number of steps between the source node and the sink may increase, this has caused the packet delivery rate of the proposed routing protocol to increase and decrease, respectively, compared to the routing protocol based on the particle swarm cluster.

Also based on the simulation results, the proposed Meta-Heuristic Moth-Flame algorithm has demonstrated its superiority over other algorithms such as Memetic Algorithm [70], HMBCR Algorithm [71], IDTOMHR Algorithm [72], in terms of several evaluation criteria. The evaluation criteria used include average transmission delay, packet delivery rate, standard deviation of energy consumption, lifetime of gateways and sensors. These results suggest that the Moth-Flame algorithm may be a promising approach for efficient and effective cluster-based routing protocols in wireless sensor networks. However, it is important to note that the comparison was made under specific scenarios and assumptions, and further research may be needed to verify the robustness and generalizability of the Moth-Flame algorithm in other settings.

4.5 Recommendations

Based on the findings of the study, some recommendations for future research in the field of wireless sensor networks can include:

1. Conducting further studies to evaluate the performance of the Moth-Flame algorithm in different types of wireless sensor networks and under various conditions.
2. Investigating the use of hybrid approaches, combining the advantages of multiple algorithms and routing protocols, to improve the performance and overcome the drawbacks of the proposed cluster-based routing protocol.
3. Developing new algorithms and routing protocols that take into account the trade-offs between network lifetime, energy consumption, and message delivery rate.
4. Exploring the impact of other factors such as network topology, traffic pattern, and node density on the performance of routing protocols and meta-heuristic algorithms in wireless sensor networks.
5. Conducting experimental studies to validate the performance of the proposed algorithms and routing protocols in real-world wireless sensor networks.
6. Investigating the scalability of the proposed algorithms and routing protocols for large-scale wireless sensor networks.

7. Examining the robustness of the proposed algorithms and routing protocols in the presence of node failures and dynamic network conditions.
8. Investigating the security aspects of the proposed algorithms and routing protocols in wireless sensor networks.

Overall, the study highlights the importance of considering the trade-offs between different performance metrics and the need for further research to develop efficient and effective algorithms and routing protocols for wireless sensor networks.

REFERENCES

[1] K. THYAGARAJAN and T. BHASKARA REDDY, “Multi-level energy efficient improved unequal clustering in wireless sensor networks & nbsp” INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING &amp; TECHNOLOGY, vol. 10, no. 2, 2019.

[2] R. Senthil Kumaran and G. Nagarajan, “Mobile sink and fuzzy based relay node routing protocol for network lifetime enhancement in Wireless Sensor Networks,” Wireless Networks, vol. 28, no. 5, pp. 1963–1975, 2022.

[3] P. Kuila and P. K. Jana, "Energy efficient clustering and routing algorithms for wireless sensor networks: Particle swarm optimization approach," Engineering Applications of Artificial Intelligence, Elsevier, Vol. 33, pp. 127-140, 2014.

[4] S. Loganathan and J. Arumugam, “Energy efficient clustering algorithm based on particle swarm optimization technique for wireless sensor networks,” Wireless Personal Communications, vol. 119, no. 1, pp. 815–843, 2021.

[5] J. Patel and H. El-Ocla, “Energy Efficient Routing Protocol in sensor networks using genetic algorithm,” Sensors, vol. 21, no. 21, p. 7060, 2021.

[6] S. Hao, Y. Hong, and Y. He, “An energy-efficient routing algorithm based on greedy strategy for energy harvesting wireless sensor networks,” Sensors, vol. 22, no. 4, p. 1645, 2022.

[7] P. Maratha and K. Gupta, “HFLBSC: Heuristic and fuzzy based load balanced, Scalable Clustering Algorithm for wireless sensor network,” 2021.

[8] J. Amutha, S. Sharma, and S. K. Sharma, “An energy efficient cluster-based hybrid optimization algorithm with static sink and mobile sink node for wireless sensor networks,” Expert Systems with Applications, vol. 203, p. 117334, 2022.

[9] S. Gupta and N. Marriwala, “Improved Distance Energy based leach protocol for cluster head election in Wireless Sensor Networks,” 2017 4th International Conference on Signal Processing, Computing and Control (ISPCC), 2017.

[10] K. Nitesh and P. K. Jana, “Convex hull-based trajectory design for mobile sink in Wireless Sensor Networks,” International Journal of Ad Hoc and Ubiquitous Computing, vol. 30, no. 1, p. 26, 2019.

[11] J. Li, Q. Han, and W. Wang, “Characteristics analysis and suppression strategy of Energy Hole in wireless sensor networks,” Ad Hoc Networks, vol. 135, p. 102938, 2022.

[12] K. Sakthivel, R. Ganesan, G. T. Devi, and M. Sowmya, “Modified Distributed Energy Efficient Clustering Protocol for lifetime maximization in Wireless Sensor Networks,” International journal of health sciences, pp. 9481–9490, 2022.

[13] S. K. Chaurasiya, S. Ghosh, and R. Banerjee, “Metaheuristic clustering in wireless sensor networks,” Computational Intelligence for Wireless Sensor Networks, pp. 37–62, 2022.

[14] S. A. Mirjalili, "Moth-Flame Optimization Algorithm: A Novel Nature-inspired Heuristic Paradigm," Elsevier, Knowledge-Based Systems Journal, Vol. 89, November 2015, pp. 228-249.

[15] A. Dahane and N.-E. Berrached, “Wireless Sensor Networks: A survey,” Mobile, Wireless and Sensor Networks, pp. 1–24, 2019.

[16] B. Dalal and S. Kukarni, “Wireless Sensor Networks: Applications,” Wireless Sensor Networks - Design, Deployment and Applications, 2021.

[17] N. Wang, Y. Zhou and W. Xiang, "An Energy Efficient Clustering Protocol for Lifetime Maximization in Wireless Sensor Networks," In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016.

[18] A. T and P. R, “System lifetime and performance improvement in wireless sensor networks by having improved energy distributed unequal Clustering Protocol,” 2021.

[19] F. Zhu and J. Wei, “An energy-efficient unequal clustering routing protocol for Wireless Sensor Networks,” International Journal of Distributed Sensor Networks, vol. 15, no. 9, p. 155014771987938, 2019.

[20] R. Govardanagiri and V. S, “Red deer and simulation annealing optimization algorithm-based energy efficient clustering protocol for improved lifetime expectancy in wireless sensor networks (wsns),” 2021.

[21] S. Nanthini, S. Nithya Kalyani, and S. Sengan, “Energy efficient clustering protocol to enhance network lifetime in wireless sensor networks,” Computers, Materials & Continua, vol. 68, no. 3, pp. 3595–3614, 2021.

[22] U. S. Verma and N. Gupta, “Wireless sensor network path optimization using sensor node coverage area calculation approach,” Wireless Personal Communications, vol. 116, no. 1, pp. 91–103, 2020.

[23] Y. El Assari, “Energy-efficient multi-hop routing with unequal clustering approach for wireless sensor networks,” SSRN Electronic Journal, 2020.

[24] A. Ishtiaq, S. Ahmed, M. F. Khan, F. Aadil, M. Maqsood, and S. Khan, “Intelligent clustering using moth flame optimizer for vehicular ad hoc networks,” International Journal of Distributed Sensor Networks, vol. 15, no. 1, p. 155014771882446, 2019.

[25] A. Pandey, A. Rajan, A. Nandi, and V. E. Balas, “Lifetime enhancement of sensor networks by the Moth Flame Optimization,” Wireless Personal Communications, vol. 118, no. 4, pp. 2807–2820, 2021.

[26] B. Ramachandra and T. P. Surekha, “Secure cluster-based routing using improved moth flame optimization for wireless sensor networks,” International Journal of Intelligent Engineering and Systems, vol. 15, no. 4, 2022.

[27] C. Trinh, B. Huynh, M. Bidaki, A. M. Rahmani, M. Hosseinzadeh, and M. Masdari, “Optimized fuzzy clustering using moth-flame optimization algorithm in wireless sensor networks,” Artificial Intelligence Review, vol. 55, no. 3, pp. 1915–1945, 2021.

[28] D. K. Kotary and S. J. Nanda, “Distributed robust data clustering in wireless sensor networks using diffusion moth flame optimization,” Engineering Applications of Artificial Intelligence, vol. 87, p. 103342, 2020.

[29] N. Mahakalkar, “Energy-efficient lifetime maximization clustering approach for ‘Wireless Sensor Networks’: A survey,” Bioscience Biotechnology Research Communications, vol. 13, no. 14, pp. 173–176, 2020.

[30] P. Bose, “Moth-flame optimization algorithm for efficient cluster head selection in wireless sensor networks,” International Journal of Swarm Intelligence Research, vol. 13, no. 1, pp. 1–14, 2022.

[31] M. Ghahramani and A. Laakdashti, “Efficient energy consumption in wireless sensor networks using an improved differential evolution algorithm,” 2020 10th International Conference on Computer and Knowledge Engineering (ICCKE), 2020.

[32] Z. Huang, “A hyper-heuristic framework for lifetime maximization in wireless sensor networks with a mobile sink,” 2020

[33] A. Jukuntla and V. Dondeti, “Energy efficiency scheme for wireless sensor network with Rendezvous nodes and Mobile Sink,” 2021 IEEE 7th International Conference on Computing, Engineering and Design (ICCED), 2021.

[34] B. Dhanalakshmi, L. SaiRamesh, and K. Selvakumar, “Intelligent energy-aware and secured QoS routing protocol with Dynamic Mobility Estimation for wireless sensor networks,” Wireless Networks, vol. 27, no. 2, pp. 1503–1514, 2021.

[35] J. Kim, Y. Kim, and Y. Yoo, “Energy-efficient routing using genetic algorithm in cluster-based wireless sensor networks,” 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), 2019.

[36] I. Baig, N. Ul Hasan, P. Valsalan, and M. Zghaibeh, “Energy Efficient Multicast Communication in Cognitive Radio Wireless Mesh Network,” Sensors, vol. 22, no. 15, p. 5601, 2022

[37] J. Dev and J. Mishra, “Energy-efficient object detection and tracking framework for Wireless Sensor Network,” Sensors, vol. 23, no. 2, p. 746, 2023

[38] N. Roy and P. Chandra, “Energy efficient positioning of base station in clustered WSN,” SSRN Electronic Journal, 2019.

[39] R. Sinde, F. Begum, K. Njau, and S. Kaijage, “Refining Network lifetime of wireless sensor network using energy-efficient clustering and DRL-based sleep scheduling,” Sensors, vol. 20, no. 5, p. 1540, 2020.

[40] J. Ramis-Bibiloni and L. Carrasco-Martorell, “Energy harvesting effect on the sensors battery lifespan of an energy efficient smartban network,” 2021 International Wireless Communications and Mobile Computing (IWCMC), 2021.

[41] S. badr, K. Aly, and A. Ghuniem, “A novel improvement in leach protocol for decreasing energy loss in wireless sensor networks,” Port-Said Engineering Research Journal, 2021.

[42] S. Suganthi, N. Umapathi, M. Mahdal, and M. Ramachandran, “Multi swarm optimization-based clustering with Tabu Search in Wireless Sensor Network,” Sensors, vol. 22, no. 5, p. 1736, 2022.

[43] V. S. Badiger and T. S. Ganashree, “Data Aggregation Scheme for IOT based wireless sensor network through Optimal Clustering Method,” Measurement: Sensors, vol. 24, p. 100538, 2022.

[44] M. Wang and J. Zeng, “Hierarchical clustering nodes collaborative scheduling in wireless sensor network,” IEEE Sensors Journal, vol. 22, no. 2, pp. 1786–1798, 2022.

[45] Y. Song, W. Qi, W. Zhao, and W. Cheng, “Full-duplex MAC protocol for CSMA/CA-based single-hop wireless networks,” Sensors, vol. 19, no. 10, p. 2413, 2019.

[46] K. Sawicki, G. Bieszczad, and Z. Piotrowski, “StegoFrameOrder—MAC layer covert network channel for wireless IEEE 802.11 networks,” Sensors, vol. 21, no. 18, p. 6268, 2021.

[47] P. B. Hari and S. N. Singh, “Security attacks at MAC and network layer in wireless sensor networks,” Journal of Advanced Research in Dynamical and Control Systems, vol. 11, no. 12, pp. 82–89, 2019.

[48] J. Park, Y. Kim, G. Kim, and H. Lim, “Majority voting-based MAC protocol for exploiting link-layer diversity in wireless networks,” Sensors, vol. 21, no. 8, p. 2706, 2021.

[49] S. Das, S. Bhowmik, and C. Giri, “Cross-layer MAC protocol for Semantic Wireless Sensor Network,” Wireless Personal Communications, vol. 120, no. 4, pp. 3135–3151, 2021.

[50] M. Lewandowski and B. Płaczek, “An event-aware cluster-head rotation algorithm for extending lifetime of wireless sensor network with Smart Nodes,” Sensors, vol. 19, no. 19, p. 4060, 2019.

[51] P. L. Nguyen, T. H. Nguyen, and K. Nguyen, “A path-length efficient, low-overhead, load-balanced routing protocol for maximum network lifetime in wireless sensor networks with holes,” Sensors, vol. 20, no. 9, p. 2506, 2020.

[52] A. Bagheri and A. Ebrahimzadeh, “Statistical analysis of Lifetime in Wireless Cognitive Sensor Network for multi-channel cooperative Spectrum Sensing,” IEEE Sensors Journal, vol. 21, no. 2, pp. 2412–2421, 2021.

[53] H. Ayadi, A. Zouinkhi, T. Val, A. van den Bossche, and M. N. Abdelkrim, “Network lifetime management in Wireless Sensor Networks,” IEEE Sensors Journal, vol. 18, no. 15, pp. 6438–6445, 2018.

[54] E. M. Mohamed, S. Hashima, A. Aldosary, K. Hatano, and M. A. Abdelghany, “Gateway selection in millimeter wave UAV wireless networks using multi-player multi-armed bandit,” Sensors, vol. 20, no. 14, p. 3947, 2020.

[55] S.-H. Choi, Y. Jang, H. Seo, B. I. Hong, and I. Ryoo, “An effective algorithm to find a cost minimizing gateway deployment for node-replaceable wireless sensor networks,” Sensors, vol. 21, no. 5, p. 1732, 2021.

[56] M. Bilal, E. U. Munir, and F. K. Alarfaj, “Hybrid clustering and routing algorithm with threshold-based data collection for heterogeneous wireless sensor networks,” Sensors, vol. 22, no. 15, p. 5471, 2022.

[57] J. Wen, “Lattice reduction and its applications in wireless sensors network,” Encyclopedia of Wireless Networks, pp. 1–4, 2018.

[58] W. M. Kempa, “Analytical model of a wireless sensor network (WSN) node operation with a modified threshold-type energy saving mechanism,” Sensors, vol. 19, no. 14, p. 3114, 2019.

[59] M. Lewandowski and B. Płaczek, “Data transmission reduction in wireless sensor network for spatial event detection,” Sensors, vol. 21, no. 21, p. 7256, 2021.

[60] C. Shi and C. Shan, “The ratio model between throughput and delay based on payload transmission time in wireless blockchain network,” Scientific Reports, vol. 12, no. 1, 2022.

[61] R. Natarajan, G. Megharaj, A. Marchewka, P. B. Divakarachari, and M. R. Hans, “Energy and distance based multi-objective red fox optimization algorithm in Wireless Sensor Network,” Sensors, vol. 22, no. 10, p. 3761, 2022.

[62] L. Huang, “Energy harvesting sensor node scheduling in wireless sensor networks,” Encyclopedia of Wireless Networks, pp. 411–414, 2020.

[63] J. Wang, Y. Gao, W. Liu, A. K. Sangaiah, and H.-J. Kim, “Energy efficient routing algorithm with mobile sink support for Wireless Sensor Networks,” Sensors, vol. 19, no. 7, p. 1494, 2019.

[64] A. Dumka, S. K. Chaurasiya, A. Biswas, and H. L. Mandoria, “Fault tolerance in wireless sensor networks,” A Complete Guide to Wireless Sensor Networks, pp. 235–249, 2019.

[65] P. Blaskiewicz, J. Cichon, M. Gebala, and M. Zawada, “Revised reliable transmission with minimal delay in wireless sensor networks,” 2020 The 9th International Conference on Networks, Communication and Computing, 2020.

[66] Y.-ki Hatada and T. Fujii, “Receiver beacon transmission interval design using Q-learning focused on packet delivery rate for Multi-Stage Wireless Sensor Networks,” 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2020.

[67] F. AL-Baadani, S. Yousef, L. AL-Jabouri, and S. Bhart, “Standard deviation based weighted clustering algorithm for wireless sensor networks,” 2016 8th Computer Science and Electronic Engineering (CEEC), 2016.

[68] R. Behrouz, “Increasing network lifetime in cluster based wireless sensor networks via Fuzzy Logic,” 2021.

[69] M. Kim, S. Park, and W. Lee, “Energy and distance-aware hopping sensor relocation for wireless sensor networks,” Sensors, vol. 19, no. 7, p. 1567, 2019.

[70] V. K. Chawra and G. P. Gupta, “Load balanced node clustering scheme using improved memetic algorithm based meta-heuristic technique for Wireless Sensor Network,” Procedia Computer Science, vol. 167, pp. 468–476, 2020.

[71] S. Al-Otaibi, A. Al-Rasheed, R. F. Mansour, E. Yang, G. P. Joshi, and W. Cho, “Hybridization of metaheuristic algorithm for Dynamic Cluster-based routing protocol in wireless sensor networksx,” IEEE Access, vol. 9, pp. 83751–83761, 2021.

[72] M. M. Asiri, S. S. Alotaibi, D. H. Elkamchouchi, A. Sayed A. Aziz, M. Ahmed Hamza, A. Motwakel, A. Sarwar Zamani, and I. Yaseen, “Metaheuristics enabled clustering with routing scheme for wireless sensor networks,” Computers, Materials &amp; Continua, vol. 73, no. 3, pp. 5491–5507, 2022.

.

PUBLICATIONS FROM THE THESIS

Conference Papers

1. Ruwaida Jasim Mamoori, and Hasan Hüseyin Balık, " Cluster-based routing by using MFO Meta-Heuristic Algorithm," ORAL SUMMARY PRESENTATION, presented at the 5th International Congress on Engineering Sciences and Multidisciplinary Approaches,pp:620, 25-26 FEBRUARY, 2023. ISTANBUL, Turkey.