Adaptive Clustering for Vehicular Networks

Thesis Submitted for Partial Fulfillment
of the Requirements for the Award of the Degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

Pawar Ayush Anjuman
Roll Number: 1901137

Under the Supervision of

Dr. Kukil Khanikar

Department of Electronics and Communication Engineering
Indian Institute of Information Technology Guwahati

30th *April, 2023*

Contents

Certificate of Approval

This is to certify that the thesis entitled **"Adaptive Clustering for Vehicular Communication"** submitted by **Pawar Ayush Anjuman** to Indian Institute of Information Technology, Guwahati, is a record of bonafide research work under my supervision and I consider it worthy of consideration for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering at Indian Institute of Information Technology Guwahati.

Dr. K u k i l   K h a n i k a r
Assistant Professor
Department of Electronics and Communication
Indian Institute of Information Technology, Guwahati
Guwahati 781001, Assam, India.

Declaration

I declare that

1. The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.

2. The work has not been submitted to any other Institute for any degree or diploma.

3. I have followed the guidelines provided by the Institute in writing the thesis.

4. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

5. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

6. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Pawar Ayush Anjuman

Abstract

Many lives are lost in road accidents. Vehicular communications are smart solutions for preventing accidents on roads, such network of communication are recognized as vehicular ad hoc networks (VANETs). VANETs are sub-class of MANETs called as Mobile ad hoc networks. VANETs differ from MANETs as the nodes travel in very high speed and change their topology rapidly, thus consistent communication among nodes become difficult. Clustering in VANETs is one of the schemes used to make topology less dynamic by assigning a cluster head, that controls and communicates among other cluster head on behalf of whole cluster, hence making the whole network topology less dynamic.

In this paper, we discuss about creating optimum number of clusters for whole network. A clustering Algorithm is proposed that considers important parameters like Velocity, Distance, and Transmission Range in consideration for creation of clusters. Then, a novel nature-inspired optimization heuristic paradigm called Moth Flame Optimization (MFO) proposed by Seyedali Mirjalili in 2015 will be used to generate optimized clusters for robust transmission in VANET Scenario. This Algorithm is used to get global optimum, by using exploration and exploitation in search space to avoid local optima and generate global optimum solution as, in such type of problems, trapping in local optima is a continuous problem.

Keywords: Vehicular networks, V2V, clustering schemes in VANET, moth-flame optimization (MFO), swarm-based algorithm, vehicular ad hoc networks (VANETs), CH election

I.   INTRODUCTION

Recent advances in wireless networks have led to the introduction of a new type of networks called vehicular ad hoc networks (VANETs). This type of networks has recently drawn significant research attention since it provides the infrastructure for developing new systems to enhance safe driving Experience. Equipping vehicles with various kinds of sensing devices and wireless communication capabilities help drivers to acquire real-time information about road conditions allowing them to react on time. For example, warning messages sent by vehicles involved in an accident enhances traffic safety by helping the approaching drivers to take proper decisions before entering the crash dangerous zone. Moreover, information about the current transportation conditions facilitate driving by taking new routes in case of congestion, thus saving time and adjusting fuel consumption. In addition to safety concerns, VANET can also support other non-safety applications that require a quality of service (QoS) guarantee.

Using all the sensors, a Vehicle can be used to communicate among the network of vehicles. But Vehicle to Vehicle (V2V) communication between nodes becomes difficult. So a technique called clustering is used. Where, vehicles with similar parameters are kept together to form a cluster and communicate among them. There will be one such node that communicates with other clusters/CHs, and such node is called Cluster Head (CHs), and all the other nodes in the cluster will be called cluster members (CMs).

Section II. Explains the motivation behind the work in our Interest in VANETs,
Section III. Explains the Objective of our Research, which explains the Research gaps in Existing Literature and the proposed solution in this paper,

Section IV. Part 1 introduces with the existing work in literature and their review. It further, briefly explains the Novel Nature Based Algorithm: MFO. Section IV. Part 2 explains our proposed algorithm with the used parameters and cost function for the algorithm. Section IV. Part 3 shows the implementation and Results for the algorithm.
Section V. concludes the work and advocates numerous guidelines for future analysis.

II.   MOTIVATION

According to the Global Status Report of 2018 on safety of Roads [1] by WHO, road accidents are the major reasons for more than 1.35 million deaths each year globally, Between 20 and 50 million more people suffer non-fatal injuries, with many incurring a disability as a result of their injury. The present traffic system has many problems. It has accident risks, problem of congestion. It is more time consuming to reach a specific destination and the travel cost is higher. So, to avoid all these issues, VANETs are the decisive solution. VANETs promises a better future to us by reducing a risk of accidents, easier/alternate route finder, overcoming problems of congestion. Hence, saving time, money & life. Nowadays, vehicles are equipped with intelligent devices and sensors to measure various attributes including location (GPS), speedometer, various computing devices, Antenna etc. These devices can be the attributes to be used in a smart vehicle. These smart Vehicles can be used to communicate with other vehicles and infrastructure to transfer information within a VANETs scenario. These vehicles collect various information like Traffic Info, Road Conditions, Accident Alert, etc. and then send it to desired units. But due to limitations of various factors like high speed, transmission range, it is not an easy task to direct information to desired units. So Clustering is used to improve VANET communication. VANETs growth will rise in the future and it is currently considered a critical research area. Many Clustering Techniques are already available in the literature [6] [7]. The main problem in forming a cluster is to form a stable cluster rather than just forming cluster. A stable cluster means a cluster that remains for comparably long duration, or having rare or less changes in Cluster Heads (CHs). Therefore, selecting the most appropriate node as the CH can ensure a stable cluster. Minimizing this switching ensures formation of stable clusters.

The No Free Lunch (NFL) Theorem [3] declares that no single technique can handle all kinds of optimization problems simultaneously. Therefore, an optimizer may accomplish decent results in a scenario while may fail in another.

MFO has been employed to solve numerous problems in literature. It is a highly effective method in solving optimization problems [7]. We will use Moth Flame Optimization [7] to relate with our Scenario to optimize the algorithm with our VANET situation.
The Proposed paper bridges the gaps in original CAMONET [4] Algorithm about its clustering criterion and proposes a method to create stable clusters. Objective/Fitness Function is created such that it takes the important parameters into consideration. The Fitness Function is Modified and used as per VANET scenario.

III.   OBJECTIVE

Our Main Objective for this paper is to deal with various aspects that affect stability of a cluster in a VANET scenario. In an existing algorithm "CAMONET: Moth-Flame Optimization (MFO) Based Clustering Algorithm for VANETs" [4] proposed in 2018. The fitness Function is defined as a weighted sum of basic parameters like relative distance among nodes and cluster head, with a Deviation in Delta Degree which is basically Deviation from ideal degree of cluster (Nodes in a Cluster) of existing cluster.

The main Drawback of this algorithm is that, in its cost function, we don't know what the exact degree of deviation would be, i.e. we can't set that ideal number of nodes in a cluster preemptively. And Apart from these, the cluster formation criterion is not described properly.

SOLUTION:

Since the main goal of clustering is to make global topology less dynamic, we believe that, changes in the network topology on the global scale are directly related to the stability of local clustering structure. Therefore, in order to enhance their stability, clustering models need to be redefined so that they are characterized based on the full status elements: speed difference, location, and direction rather than considering only position and Delta degree deviation.

Our Field of interest is to implement updated fitness/ Cost function with respect to other important factor in VANET like relative velocity of cluster head to its cluster members. This ensures us even better and more Stable cluster formation. Apart from these, we also consider, Relative Distance between the nodes which can be derived from Euclidean distance between two nodes.

Apart from the Cost Function, we have also computed the method to form stable clusters.

IV. WORK DONE

1 RELATED WORK / LITERATURE REVIEW

VANETs stands for Vehicular ad-hoc networks. The Group of nodes are connected wirelessly, i.e. there is no fixed hardware infrastructure, and the nodes communicates with other nodes to form a wireless network. The Topology in VANETs is ever changing, it changes abruptly and arbitrarily. Such Networks then contribute to a bigger Network, Various Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) are used to communicate between the networks. VANETs are foreseen by the automotive business as one of the most vital upcoming technologies. It can sustain a huge number of applications comprising of traffic management, safety and infotainment. The task of transferring data becomes tremendously challenging. To tackle all these issues and create a reliable and efficient network, vehicles are grouped into clusters. Clustering creates dispersed structure of categorized network arrangements by grouping nodes having similar properties with each other like relative velocity and co-related spatial distribution.
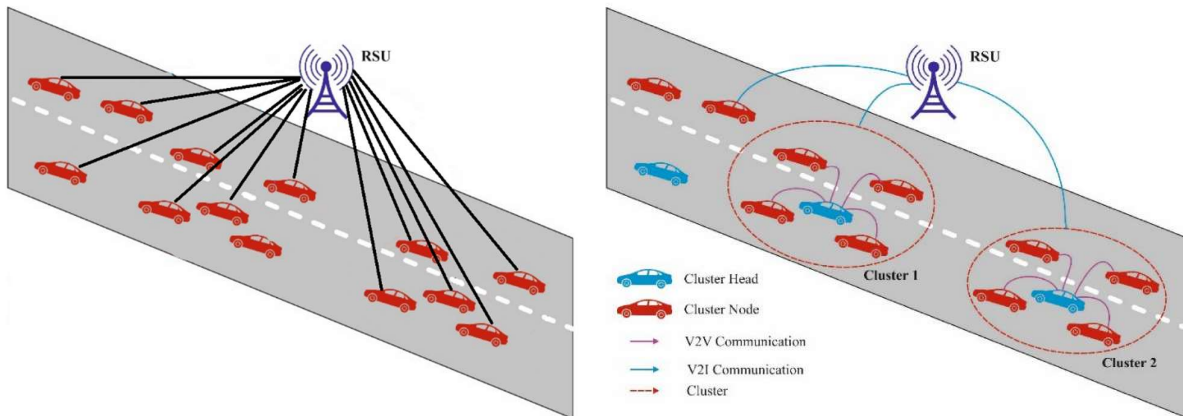
For efficient networking in VANETs there should be reduced number of CHs and lifetime of clusters should be longer. The Method to improve the performance by using limited number of resources is known as optimization. Most of the algorithms were modelled mathematically before the emergence of heuristic optimization algorithms. These mathematical optimization approaches have a severe problem of local optima stagnation [7]. Local Optima Stagnation is a term used where an algorithm get stuck in a Local Optima rather than the global Optima, this is due to a reason that clustering in VANETs is considered to be a NP-Hard Problem[26] due to its nature of ever changing topology. Therefore, these methods are not helpful in solving real world problems. To find, near optimal solution, swarm based optimization can be applied. Swarm intelligence procedures are founded on the collection of animals such as birds, ants, moths known as a swarm. This becomes our motivation for using Moth based Moth-Flame

optimization (MFO) for optimizing our algorithm.

ACO, an algorithm based on movement of ants, proposed in 2016, Ants uses pheromone (a chemical substance) trail while moving. The following ants uses the trail of pheromone to track the route. The concentration of pheromone decides the route to be chosen to travel, as more number of ants travels on the route the concentration increases. Hence, the concentration of pheromone decides the path for the later ants thus achieving indirect communication between the ants. These techniques have been widely used for solving complex optimization problems

1.1 CONVENTIONAL MOTH FLAME OPTIMIZATION (MFO)

A. MOTIVATION:

Moths are small butterfly like bugs that are present in nature. The Motivation behind this algorithm is the movement pattern of moths at night. Moths follow moon light to travel in night. Moths smartly uses moon to travel by keeping a constant angle to the moon so as to travel in a straight line in long distances. This phenomenon of moths to travel is called as transverse orientation, But sometimes, these moths confuses man-made lights with moon and keep a same constant angle with the manmade lights. However, these light sources are way too close with respect to moon so moths often causes to trap themselves into a deadly lethal spiral, as maintaining same angle causes a moth to fly spirally. These transverse orientation becomes the motivation of this algorithm, we use exploration and exploitation to explore the search space and find global optima avoiding local optima. As this Algorithm is population based, we can avoid local optimum.
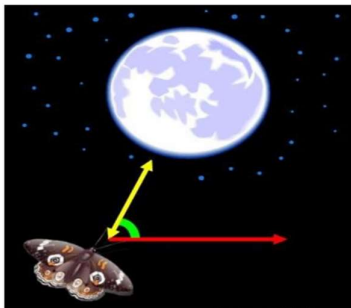

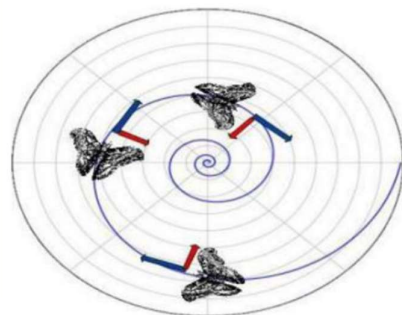Fig: Transverse Orientation   Fig: Moths Flying in Spiral manner along Streetlight   Fig: Spiral Flying Path

B. MFO ALGORITHM

The candidate solutions in the proposed MFO Algorithm are assumed to be moths, and the variables in the problem are the locations of the moths in the space. The moths may therefore change their location vectors and fly in 1-D, 2-D, 3-D, or hyper dimensional space. The population-based nature of the MFO method requires that the set of moths(M) be represented in a matrix as follows:

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & \cdots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & \cdots & m_{n,d} \end{bmatrix}$$

Where, n is the number of moths and d is the number of variables (dimension).
For all the moths, we also assume that there is an array for storing the corresponding fitness values as follows:

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

Where n is the number of moths.

The fitness value is the cost function's return value for each moth. Each moth's position vector is input into the fitness function as rows in the matrix M, and the output of the fitness function is assigned to the corresponding moth as its fitness value, such as OM1 in the matrix OM. Flames are yet another essential element of the suggested method. The following is an example of a matrix resembling the moth matrix:

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \cdots & \cdots & F_{1,d} \\ F_{2,1} & F_{2,2} & \cdots & \cdots & F_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ F_{n,1} & m_{n,2} & \cdots & \cdots & F_{n,d} \end{bmatrix}$$

Where n is the number of moths.

Moths and flames are both Solutions. The way we handle and update them in each iteration makes a distinction between them. The ideal position for moths to date has been in flames, which are actual search agents that travel around the search area. In other words, as moths are searching the search area, flames can be thought of as flags or pins that are dropped.
Each moth therefore scans the area around a flag (flame) and modifies it if a better option is discovered. A moth never loses its optimal solution with this process.

The MFO algorithm is a three-tuple that approximates the global optimal of the optimization problems and defined as follows:

MFO = (I, P, T)

I is a function that generates a random population of moths and corresponding fitness values. The function is described as follows:

I: ∅ → {M, OM}

The P function, which is the main function, moves the moths around the search space. This function received the matrix of M and returns its updated one eventually.

P : M → M

The T function returns true if the termination criterion is satisfied and false if the termination criterion is not satisfied:

T : M → {true, false}

With I, P, and T, the general framework of the MFO algorithm is defined as follows:

```
M = I();
while T(M) is equal to false
 M = P(M);
end
```

The function I has to generate initial solutions and calculate the objective function values. Any random distribution can be used in this function. The following method is utilized as the default:

```
for i = 1: n
 for j = 1: d
       M(i,j) = (ub(i) lb(i)) ⁄ rand() + lb(i);
  end
end
OM = FitnessFunction(M);
```

As can be seen, there are two other arrays called ub and lb. These matrixes define the upper and lower bounds of the variables as follows:

$$ub = [ub_1, ub_2, ub_3, \ldots, ub_{n-1}, ub_n]$$

where ubi indicates the upper bound of the i-th variable.

$$lb = [lb_1, lb_2, lb_3, \ldots, lb_{n-1}, lb_n]$$

Where lbi indicates the lower bound of the i-th variable. After the initialization, the P function is iteratively run until the T function returns true. The P function is the main function that moves the moths around the search space. As mentioned above the inspiration of this algorithm is the transverse orientation. In order to mathematically model this behavior, the position of each moth is updated with

respect to a flame using the following equation:

  Mi = S (Mi, Fj)

Where Mi indicate the i-th moth, Fj indicates the j-th flame, and S is the spiral function.

  S (Mi, Fj) = Di.e^{bt}.cos (2.pi.t) + Fj

 Where Di indicates the distance of the i-th moth for the j-th flame, b is a constant for defining the shape of the logarithmic spiral, and t is a random number in [-1, 1].

 D is calculated as follows:

 Di = |Fj – Mi|

 Where, Mi indicate the i-th moth, Fj indicates the j-th flame, and Di indicates the distance of the i-th Moth for the j-th flame.

2   PROPOSED ALGORITHM

2.1 MFO BASED CLUSTERING ALGORITHM

In the proposed MFO based clustering algorithm, we assume that Moth matrix consists of six different dimensions/parameters where the position$_x$ and position$_y$ are 'x' and 'y' coordinates of moth's position. Whereas the velocity being the third parameter, velocity is taken in 1D as vehicles will only travel either in straight direction or opposite direction, consecutively + or - sign represents its direction, The connected nodes i.e. Nodes in the Transmission Range serves as the fourth dimension in the moth matrix, The rest two dimensions are consecutively summation of relative distances and summation of Relative Velocities. So, our Moth Matrix would be represented as,

$$
M = \begin{bmatrix}
d_{x1} & d_{y1} & v_1 & CN_1 & \sum |d_{1j}| & \sum |v_{1j}| \\
d_{x2} & d_{y2} & v_2 & CN_2 & \sum |d_{2j}| & \sum |v_{2j}| \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
d_{xn} & d_{yn} & v_n & CN_n & \sum |d_{nj}| & \sum |v_{nj}|
\end{bmatrix}
$$

 Additionally, we compute Distance Matrix corresponding to Distance between each nodes, where

$$D_{ij} = abs (E.D. (Node_i, Node_j));$$

Where,   E.D. represents Euclidean Distance between Node$_i$ and Node$_j$
Abs represents Absolute Value of the output.
D$_{ij}$ represents element of Distance Matrix.

$$D = \begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1n} \\ D_{21} & D_{22} & \cdots & D_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ D_{n1} & D_{2n} & \cdots & D_{nn} \end{bmatrix}$$

Now, we compute the velocity matrix similar to D matrix that represents the relative velocity between two Nodes.

$$V_{ij} = abs\ (Velocity_{Nodei} - Velocity_{Nodej})$$

Where,   Abs represents Absolute Value of the output.
V$_{ij}$ represents element of Velocity Matrix.

$$V = \begin{bmatrix} V_{11} & V_{12} & \cdots & V_{1n} \\ V_{21} & V_{22} & \cdots & V_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ V_{n1} & V_{2n} & \cdots & V_{nn} \end{bmatrix}$$

Now, we check if the relative distance between two nodes is in transmission range, if it exceeds the transmission range set the D$_{ij}$ and V$_{ij}$ values to zero and count such nodes, then count the number of nodes in transmission range using (N-count) where N represents Total No. of Nodes, and count represents No. of Nodes that exceeds the transmission Range. And set it to column no. 4 of moth matrix.

We then generate the sum of all the relative velocities and relative Distances of each row of Velocity Matrix and Distance Matrix Respectively, and set it to column no. 6 and 5 respectively.

We, then feed these Moth Matrix to MFO algorithm, to find the best flames in order to achieve the most fit and stable value for making cluster head, once we get these value, we then check the no. of nodes that are connected to it. After that we check for the second fit value and declare that node as second cluster's head, and so on until the whole network is connected. Hence we get the total number of clusters required to cover the network.

The Code is attached in the appendix section of this Document for Reference.

2.2 THE FITNESS/COST FUNCTION

As Discussed in the previous version of this document, the algorithm can be derived for multi-objective optimization, where weights can be assigned to each objective according to the conditions preferred by user. Here the fitness function is also a weighted function. We consider two parameters as important variables defined as relative distance, relative velocity and the nodes connected to it. The Fitness function can be derived as:

$$f_{obj} = f(F_1, F_2, F_3)$$

$$f_{obj} = (F_1 * W_1 + F_2 * W_2) / F3$$

Where,

f_{obj} is an objective function of F_1, F_2, F_3 and W_1 and W_2 are weights of objective functions respectively.

Both the weights have been assigned similar value i.e., 0.5, however these values can be modified in accordance with the user's preference.

F_1 Function represents the sum of relative Distance of nodes in a cluster w.r.t Cluster Head.

F_2 Function represents the sum of relative velocity of nodes in a cluster w.r.t Cluster Head.

F_3 Function represents number of nodes in a cluster.

In our case, we computed F_1 already in our moth matrix as its 5th dimension and F_2 as its 6th dimension.

3. IMPLEMENTATION AND RESULTS

3.1 IMPLEMENTATION

In this section, experimental setup is depicted alongside a correlation of the outcomes for our accomplished investigations. The Results can be verified by changing Transmission Range and increasing the grid size.

3.2 EXPERIMENTAL SETUP

The Experiments are accomplished using MATLAB online. The Experiments are performed on variable grid sizes of 1 km x 1km to 4 km x 4km network size. The Transmission Range is also altered from 50 m to 500 m. The nodes move along the X-axis in a bi-directional manner. The speed of nodes is consistently altered from 80 km/h to 120 km/h.

3.3   RESULTS

A.   Effect of Increasing the Transmission Range.

```
Total Number of Search Agents are : 100
Transmission Range is : 50m
The Grid size is : 1000 m x 1000 m
Optimizing Your Problem...
Total No. of Optimum cluster is 28
>>
```

Figure 1

```
Total Number of Search Agents are : 100
Transmission Range is : 100m
The Grid size is : 1000 m x 1000 m
Optimizing Your Problem...
Total No. of Optimum cluster is 9
>>
```

Figure 2

```
>> main
Total Number of Search Agents are : 100
Transmission Range is : 200m
The Grid size is : 1000 m x 1000 m
Optimizing Your Problem...
Total No. of Optimum cluster is 3
>>
```

Figure 3

```
Total Number of Search Agents are : 100
Transmission Range is : 500m
The Grid size is : 1000 m x 1000 m
Optimizing Your Problem...
Total No. of Optimum cluster is 2
>>
```

Figure 4

We can see from the above Results in Figure 1, 2, 3 and 4 that as the transmission range is increased the total number of clusters reduces from 28 to 9 to 3 to 2 respectively, The Total Number of search agents is kept to 100, The Grid Size is Kept Equal in all four figures i.e. 1km x 1km. This also verifies that as Transmission range is increased in vehicles, a cluster can connect to a vehicle for a longer distance, hence connectivity is improved. This suggests the inverse relation between the number of clusters and Transmission range. This verifies our Results.

B.   Effect of Increasing the Grid Size.

```
Total Number of Search Agents are : 100
Transmission Range is : 70m
The Grid size is : 1000 m x 1000 m
Optimizing Your Problem...
Total No. of Optimum cluster is 17
>>
```

Figure 5

```
Total Number of Search Agents are : 100
Transmission Range is : 70m
The Grid size is : 2000 m x 2000 m
Optimizing Your Problem...
Total No. of Optimum cluster is 47
>>
```

Figure 6

```
Total Number of Search Agents are : 100     Total Number of Search Agents are : 100
Transmission Range is : 70m                 Transmission Range is : 70m
The Grid size is : 3000 m x 3000 m          The Grid size is : 4000 m x 4000 m
Optimizing Your Problem...                  Optimizing Your Problem...
Total No. of Optimum cluster is 64          Total No. of Optimum cluster is 77
>>                                          >>
```

Figure 7   Figure 8

We can see from the above Results in Figure 5, 6, 7 and 8 that as the Grid size is increased, the total number of clusters increases from 17 to 47 to 64to 77 respectively, The Total Number of search agents is kept to 100, The Transmission range is Kept Equal in all four figures i.e. 70 m. This also verifies that as Grid size is increased in vehicles, with same Transmission Range it becomes difficult for a node to connect to all the vehicles, hence connectivity is reduced. This suggests the direct relation between the number of clusters and Grid size. This verifies our Results.

```
Total Number of Search Agents are : 100     Total Number of Search Agents are : 100
Transmission Range is : 100m                Transmission Range is : 400m
The Grid size is : 4000 m x 4000 m          The Grid size is : 4000 m x 4000 m
Optimizing Your Problem...                  Optimizing Your Problem...
Total No. of Optimum cluster is 49          Total No. of Optimum cluster is 10
>>                                          \\
```

Figure 9   Figure 10

We can see Again in Figure 9 and 10 that as the Transmission range is increased from 100m to 400 m. the number of clusters decreases from the previous number of clusters from 49 to 10 for grid sizes 4km x 4km.

Therefore, the number of clusters required for a network are directly proportional to grid size. On the other hand, it is inversely proportional to the transmission range of the nodes. This gives us the following two equations.

$$x \propto Gs$$
$$x \propto \frac{1}{Tr}$$

Where,

x represents the total number of clusters
G_s represents the Grid Size.
T_x represents the Transmission Range.

16

V. Conclusion and Future Work

We can conclude that our proposed algorithm generates near optimal results by taking the important factors like velocity and distance that affects the stability of cluster. Our work generates Clusters using relevant parameters and then uses novel nature inspired moth-flame optimization (MFO) to get Optimum number of clusters. Owing to the evolutionary competency of our algorithm, greater search spaces are administered, and the values of objective function are modified vigorously.
We can verify with our Results that Our Algorithm is effective and flexible and produces minimum number of clusters in different network scenarios.

One can Notice the direct relation of number of clusters and Network size, and the indirect relation with Transmission range which validates the effectiveness of our Algorithm. Our Algorithm Generates Better Results even in densely Traffic Area. The Resource cost is hence diminished by decreasing the number of clusters in the network.

To extend the scope of Research in Future, list of Objectives can be enhanced, we can take important factors like SNR, Cluster lifetime, Density, etc. for getting even better results. Apart from these, a clustering method can be created for a scenario when a specific node leaves the cluster and joins other cluster. Again, if the cluster head itself leaves the node, then we can create a criterion to reform the new cluster.

Bibliography

[1]   Global Status Report on Road Safety 2018: Summary, World Health Organization, Geneva, Switzerland, 2018.

[2]   L. A. Maglaras and D. Katsaros, ''Distributed clustering in vehicular networks,'' in Proc. IEEE 8th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob), Oct. 2012, pp. 593–599.

[3]   X. Cheng and C. Huang, ''A center-based secure and stable clustering algorithm for VANETs on highways,'' Wireless Commun. Mobile Comput., vol. 2019, Jan. 2019, Art. no. 8415234.

[4]   Y. A. Shah, H. A. Habib, F. Aadil, M. F. Khan, M. Maqsood and T. Nawaz, "CAMONET: Moth-Flame Optimization (MFO) Based Clustering Algorithm for VANETs", IEEE Access, vol. 6, pp. 48611-48624, September. 2018.

[6]   D. H. Wolper and W. G. Macready, ''No free lunch theorems for optimization,'' IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 67–82, Apr. 1997.

[7]   Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems, 89, pp.228-249.

[8]   B. S. Yildiz and A. R. Yildiz, ''Moth-flame optimization algorithm to determine optimal machining parameters in manufacturing processes,'' Mater. Test., vol. 59, no. 5, pp. 425–429, 2017, doi: 10.3139/120.1110242017.

Appendix A

1. Codes

1.1 Get_Functions_details.m

```matlab
% lb is the lower bound: lb=[lb_1,lb_2,...,lb_d]
% up is the upper bound: ub=[ub_1,ub_2,...,ub_d]
% dim is the number of variables (dimension of the problem)
% we can define other objective functions for different parameters

function [lb,ub,dim,fobj] = Get_Functions_details(F)


switch F

    case 'F1'
        fobj = @F1;
        lb=[-2000, -2000, -80, 0, 0, 0];
        ub=[ 2000,  2000,  80, 0, 0, 0];
        grid_size_x=ub(1)-lb(1);
        grid_size_y=ub(2)-lb(2);
        disp(['The Grid size is : ',num2str(grid_size_x),' m x
',num2str(grid_size_y), ' m']);
        dim=6; %predefined
end

end

% F1

function o = F1(x)

w1=0.5;
w2=0.5;
o=(x(5).*w1 + x(6).*w2)/x(4);
end
```

1.2 Initialization.m

```matlab
% This function creates the first random population of moths

function X=initialization(SearchAgents_no,dim,ub,lb)

Boundary_no= size(ub,2); % numnber of boundaries

% If the boundaries of all variables are equal and user enter a single number for
both ub and lb
if Boundary_no==1
    X=rand(SearchAgents_no,dim).*(ub-lb)+lb;
end

% If each variable has a different lb and ub
if Boundary_no>1
    for i=1:dim
        ub_i=ub(i);
        lb_i=lb(i);
        X(:,i)=rand(SearchAgents_no,1).*(ub_i-lb_i)+lb_i;
    end
end
```

1.3 MFO1.m

```matlab
function [Best_flame_score,Best_flame_pos]=MFO1(N,Max_iteration,lb,ub,dim,fobj,Tx)

disp('Optimizing Your Problem...');

%Initialize the positions of moths
Moth=initialization(N,dim,ub,lb);

Scores=zeros(1,Max_iteration);

Iteration=1;
stall_iteration=0;

% Main loop
while ((Iteration<=Max_iteration) && stall_iteration<=15)

    % Number of flames
    Flame_no=round(N-Iteration*((N-1)/Max_iteration));

    %formation of dist and velocity matrix
    for i=(1:N)
        count = 0;
        for j=(1:N)
            dist(i,j)=abs(sqrt((Moth(i,1)-Moth(j,1)).^2 + (Moth(i,2)-
Moth(j,2)).^2));
            vel(i,j)=abs(Moth(i,3)-Moth(j,3));
            if dist(i,j)>2*Tx
                dist(i,j)=0;
                vel(i,j)=0;
                count=count+1;
            end
        end
        Moth(i,4)=N-count; % Connected Nodes/ Nodes in Range
    end

    for i=(1:N)
        % Calculate the fitness of moths
        Moth_fitness(1,i)=fobj(Moth(i,:));  % Generating OM Matrix

    end

    if Iteration==1
        % Sort the first population of moths
        [fitness_sorted, I]=sort(Moth_fitness);
        sorted_population=Moth(I,:);

        % Update the flames
        best_flames=sorted_population;
        best_flame_fitness=fitness_sorted;
    else

        % Sort the moths
        double_population=[previous_population;best_flames];
        double_fitness=[previous_fitness best_flame_fitness];

        [double_fitness_sorted, I]=sort(double_fitness);
```

```matlab
        double_sorted_population=double_population(I,:);

        fitness_sorted=double_fitness_sorted(1:N);
        sorted_population=double_sorted_population(1:N,:);

        % Update the flames
        best_flames=sorted_population;
        best_flame_fitness=fitness_sorted;
    end

    % Update the position best flame obtained so far
    Best_flame_score=fitness_sorted(1);
    Best_flame_pos=sorted_population(1,:);

    previous_population=Moth;
    previous_fitness=Moth_fitness;

    % a linearly dicreases from -1 to -2 to calculate t in Eq. (3.12)
    a=-1+Iteration*((-1)/Max_iteration);

    for i=1:N

        for j=1:dim
            if i<=Flame_no % Update the position of the moth with respect to its
corresponding flame

                % Dist. to flame
                distance_to_flame=abs(sorted_population(i,j)-Moth(i,j));
                b=1;
                t=(a-1)*rand+1;


Moth(i,j)=distance_to_flame*exp(b.*t).*cos(t.*2*pi)+sorted_population(i,j);
            end

            if i>Flame_no % Update the position of the moth with respct to one flame

                distance_to_flame=abs(sorted_population(i,j)-Moth(i,j));
                b=1;
                t=(a-1)*rand+1;


Moth(i,j)=distance_to_flame*exp(b.*t).*cos(t.*2*pi)+sorted_population(Flame_no,j);
            end

        end

    end

    Scores(Iteration)=Best_flame_score;
    if(Iteration>1 && (Scores(Iteration) == Scores(Iteration-1)))
        stall_iteration = stall_iteration+1;
    else
        stall_iteration = 0;
    end

    % Display the iteration and best optimum obtained so far
    if mod(Iteration,50)==0
        display(['At iteration ', num2str(Iteration), ' the best fitness is ',
num2str(Best_flame_score)]);
```

```matlab
        end
    Iteration=Iteration+1;
end

total_nodes=0;
Total_no_of_cluster=0;
for i=1:N
    if(total_nodes<N)
        total_nodes=total_nodes+best_flames(i,4);
        Total_no_of_cluster=Total_no_of_cluster+1;
    end
end
display(['Total No. of Optimum cluster is ', num2str(Total_no_of_cluster)]);
```

1.4 main.m

```matlab
clear all
clc


SearchAgents_no=100; % Number of search agents
display(['Total Number of Search Agents are : ', num2str(SearchAgents_no)]);
Function_name='F1';

Max_iteration=1000; % Maximum number of iterations
Tx=100;
display(['Transmission Range is : ', num2str(Tx),'m']);

[lb,ub,dim,fobj]=Get_Functions_details(Function_name);

[Best_score,Best_pos]=MFO1(SearchAgents_no,Max_iteration,lb,ub,dim,fobj,Tx);

% display(['The best solution obtained by MFO is : ', num2str(Best_pos)]);
```