A
Case Study On

Dynamic Clustering in WSN using MATLAB

Under the Course
"Sensor Network" (CI3061)

Prepared by
Third Year (*Information Technology Department*)

Sr. No.	Student Name	Roll. No.
1.	Sejal Kiran Hankare	2010050
2.	Sanjivani Mahadev Balte	2010059
3.	Sanika Hanmant Kolekar	2010061
4.	Rutuja Rajaram Khot	2010064

Under the Guidance of
Prof. D. T. Mane
(Information Technology Department)

K. E. Society's
Rajarambapu Institute of Technology, Rajaramnagar
(An Autonomous Institute Affiliated to Shivaji University, Kolhapur)

2022-2023

1. Introduction :-

Clustering in WSN:

Clustering is the process partitioning a group of sensor into small numbers of clusters. In environments where the sensors are mobile clusters cannot be static. Like cluster heads in each cluster are elected dynamically, the members in each cluster also need to be dynamically identified. Therefore the size of each cluster is not fixed and can vary depending on the position of the sensors. Dynamic Clustering helps in efficiently grouping sensors into clusters dynamically. There is no fixed cluster size and the sensors are divided into the required number of clusters with members of each cluster calculated dynamically.

Here's an overview of the Dynamic Clustering Algorithm:

Clustering using k-means algorithm:

K means(X,k) partitions the points in the n-by-p data matrix X into k clusters. This iterative partitioning minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid distances. Rows of X correspond to points, columns correspond to variables. K means returns an n-by-1 vector IDX containing the cluster indices of each point. By default, kmeans uses squared Euclidean distances. When X is a vector, kmeans treats it as an n-by-1 data matrix, regardless of its orientation.

The sensor positions and number of clusters,

X - a matrix containing the x, y coordinates of the sensors in the scenario

K - the number of clusters are passed to k-means algorithm.

[IDX, C] = k-means(X,k)

IDX - Contains the cluster id’s of each sensor, the cluster to which the sensorbelongs.

C - centroids of each cluster

Clustering using Fuzzy C-Means Algorithm:

Fuzzy c-means (FCM) is a data clustering technique in which a dataset is grouped into n clusters with every data point in the dataset belonging to every cluster to a certain degree. For example, a certain data point that lies close to the center of a cluster will have a high degree of belonging or membership to that cluster and another data point that lies far away from the center of a cluster will have a low degree of belonging or membership to that cluster.

2. Literature Review :-

Sr. No.	Citation	Abstract of Paper
1.	Krishnan R, Starobinski D, Message-efficient self-organization of wireless sensor networks[J], In: Proceedings of the Wireless Communications and Networks, New Orleans, Louisiana, USA, 2003	In this paper a clustering-based routing protocol that minimizes global energy usage by distributing the load to all the nodes at different points in time Distributing the energy among the nodes in the network is effective in reducing energy dissipation from a global perspective and enhancing system lifetime.
2.	Chandrakasan A, Cho S, Design Considerations for distributed micro sensor systems[C], In: Proceedings of the IEEE 1999 Custom Integrated Circuits Conference. San Diego, CA, 1999:279~286	This paper proposes a routing scheme called robust multi-path routing that establishes and uses multiple node-disjoint routes. The performance comparison of this protocol with dynamic source routing (DSR) by OPNET simulations shows that this protocol is able to achieve a remarkable improvement in the packet delivery ratio and average end-to-end delay.
3.	O. Younis, S. Fahmy, "Heed: A hybrid, energy-efficient, distributed clustering approach for ad-hoc sensor networks," IEEE Trans. On Mobile Computing, vol. 3, no. 4, pp. 660-669, Apr. 2004.	In this paper, presented a distributed, energy-efficient clustering approach for ad-hoc sensor networks. approach is hybrid: cluster heads are probabilistically selected based on their residual energy, and nodes join clusters such that communication cost is minimized. They assume quasi-stationary networks where nodes are location-unaware and have equal significance. A key feature of their approach is that it exploits the availability of multiple transmission power levels at sensor nodes.

3. Key Features of Dynamic Clustering :-

Dynamic clustering in Wireless Sensor Networks (WSNs) refers to the process of organizing sensor nodes into clusters dynamically based on certain criteria or conditions. Here are some key features of dynamic clustering in WSNs:

- Adaptive Cluster Formation: Dynamic clustering allows the formation of clusters in a self-organizing manner, where sensor nodes autonomously determine their cluster membership based on local information. The clusters can be formed and reconfigured dynamically as the network conditions change.
- Energy Efficiency: Dynamic clustering helps in achieving energy efficiency in WSNs. By forming clusters, sensor nodes can aggregate and relay data to the cluster head, which reduces the overall energy consumption of the network. Cluster heads can perform data fusion, compression, and transmit aggregated data to the base station, reducing the number of transmissions and conserving energy.
- Load Balancing: Dynamic clustering enables load balancing among the sensor nodes in the network. Cluster heads distribute the sensing and communication load among the nodes within their respective clusters, ensuring that no single node becomes overloaded. This helps in prolonging the network lifetime by evenly distributing energy consumption.
- Fault Tolerance: Dynamic clustering improves the fault tolerance of WSNs. In the event of node failures or network partitions, the dynamic clustering algorithm can adaptively reconfigure the clusters and select new cluster heads to maintain network connectivity and data transmission.
- Scalability: Dynamic clustering supports the scalability of WSNs. As the network size grows, the clustering algorithm can dynamically adjust the number of clusters and cluster heads to accommodate the increased number of sensor nodes. This allows the network to efficiently handle a large number of nodes without incurring excessive overhead.
- Data Aggregation and Fusion: Dynamic clustering facilitates data aggregation and fusion within clusters. Cluster heads collect data from member nodes, perform data fusion techniques to eliminate redundancy and reduce the data size, and transmit the aggregated data to the base station. This reduces the amount of data transmitted over the network and conserves energy.
- Network Adaptability: Dynamic clustering allows the network to adapt to changes in network conditions, such as node mobility, varying traffic patterns, and dynamic network topologies. The clustering algorithm can continuously monitor the network state and reconfigure the clusters accordingly, ensuring efficient data collection and transmission.

These are some of the key features of dynamic clustering in Wireless Sensor Networks. The specific implementation and characteristics of dynamic clustering algorithms may vary based on the requirements of the WSN application and the network environment.

4. Advantages :-

Dynamic clustering in Wireless Sensor Networks (WSNs) offers several advantages over static clustering approaches. Here are some of the key advantages:

- Energy Efficiency: Dynamic clustering helps in improving the overall energy efficiency of the network. By periodically reorganizing the clusters based on the changing network conditions, energy consumption can be optimized. Cluster heads can be dynamically selected based on their residual energy levels or other energy metrics, ensuring a balanced distribution of energy consumption among nodes.

- Load Balancing: Dynamic clustering facilitates load balancing among sensor nodes. As network conditions change, some nodes may become more heavily loaded than others due to varying data traffic patterns or environmental conditions. By re-clustering, the network can distribute the load more evenly across nodes, preventing the early exhaustion of resources in certain nodes and enhancing the network's overall performance.

- Scalability: Dynamic clustering helps in achieving scalability in WSNs. As the network size increases or new nodes are added, dynamic clustering algorithms can adapt and reconfigure the network structure accordingly. This allows the network to accommodate a larger number of nodes without significant performance degradation, ensuring the scalability of the WSN.

- Fault Tolerance: Dynamic clustering enhances the fault tolerance of WSNs. By periodically re-electing cluster heads, the network can adapt to node failures or sudden changes in network topology. If a cluster head fails, a new cluster head can be selected, and the network can be reconfigured dynamically without disrupting the overall network operation.

- Improved Network Lifetime: With better energy efficiency, load balancing, fault tolerance, and scalability, dynamic clustering helps in extending the overall network lifetime. By reducing energy consumption, evenly distributing the load, adapting to failures, and accommodating network growth, the network can operate for a longer duration before nodes run out of energy, leading to improved network sustainability.

Overall, dynamic clustering in WSNs provides numerous advantages, including enhanced energy efficiency, load balancing, scalability, fault tolerance, efficient data aggregation, and extended network lifetime. These advantages contribute to improved performance, resource utilization, and resilience in wireless sensor networks.

5. Limitations :-

Dynamic clustering in Wireless Sensor Networks (WSNs) offers several advantages such as energy efficiency, load balancing, and scalability. However, it also has certain limitations that should be taken into consideration. Some of the limitations of dynamic clustering in WSNs include:

- Overhead: Dynamic clustering requires additional control messages and communication overhead for cluster formation, maintenance, and reconfiguration. These overheads can consume significant network resources and reduce the overall efficiency of the network.

- Cluster Formation Delay: The process of dynamic cluster formation involves sensor nodes exchanging information, calculating metrics, and selecting cluster heads. This process introduces delays, especially when the network size is large or the network topology changes frequently. The delay in cluster formation can impact the real-time responsiveness of the network.

- Cluster Head Selection: The selection of cluster heads in dynamic clustering algorithms can be challenging. It often involves complex calculations based on parameters such as energy level, distance, or node density. Incorrect or suboptimal cluster head selection can result in imbalanced energy consumption or poor network performance.

- Communication Overhead: The exchange of control messages and coordination among cluster heads and cluster members can generate significant communication overhead in the network. This overhead can consume precious energy resources and affect the overall network lifetime.

It is essential to consider these limitations when designing and implementing dynamic clustering algorithms in WSNs and to select appropriate algorithms that address the specific requirements and challenges of the network.

6. Dynamic Clustering Algorithm Implementation :-

```
% Dynamic Clustering Simulation

function [A,B,C] = clustering(x, scount, num_cls, power, max_energy)
% changed clustering function. New paramter power: column vector of
% remaining power for each device
% s_count is sensor_count
% Clustering_Method = 1                    K Means using distance
%                   = 2                    Fuzzy C Means using distance
%                   = 3      K Means using distance and power
%                   = 4      Fuzzy C Means using distance and power


Clustering_Method = 1;

%savedynamic_clustering.mat
%change here for different algorithm

    if(Clustering_Method == 1 || Clustering_Method == 3)
            [IDX,C]= k_means(x,num_cls);

    else
            [IDX,C]= fuzzy(x,num_cls);
    End

       cl_count=zeros(1,num_cls);
    cl_dist=zeros(1,scount);

    if(Clustering_Method> 2)                % only when method involves power
            cl_max_dist = zeros(1,num_cls);    % max distance in each cluster
            cl_max_power = zeros(1,num_cls); % maxdevicepowerleftineachcluster
    end

% Calculation of Distance from each sensor to the centroid of its cluster and size of each cluster

    for index1=1:1:num_cls
            for index2=1:1:scount
                    if IDX(index2)==index1
                            cl_count(index1)=cl_count(index1)+1;
                            cl_dist(index2)=
pdist([C(index1),C(num_cls+index1);x(index2,1),x(index2,2)],'euclidean'); % hard coded to be modified

                            if(Clustering_Method > 2)                % only when method involves power
                                    if cl_max_dist(index1) < cl_dist(index2)
                                    cl_max_dist(index1) = cl_dist(index2);
                                    end

                                    if cl_max_power(index1) < power(index2)
                                    cl_max_power(index1) = power(index2);
                                    end
                            end
                    end
            end
    end


    cl_index = zeros(1,num_cls);
    c_head = zeros(1,num_cls);
```

```
% Cluster Head election

    for index1=1:1:num_cls
            if(Clustering_Method > 2)
                    curr_cl_max_p = cl_max_power(index1);
                    curr_cl_max_d = cl_max_dist(index1);
            end

            prev_dist_p = 100000;
            for index2=1:1:scount
             if IDX(index2)==index1
                    cl_index(index1)=cl_index(index1)+1;
                    curr_var = cl_dist(index2);

                    if(Clustering_Method> 2)  %only when method involves power

                            curr_var = (curr_cl_max_p * cl_dist(index2)/(curr_cl_max_d + 0.0001)) -
10*power(index2);
                     end

                     if(min(prev_dist_p,curr_var)== curr_var)
                            c_head(index1)=  index2;
                            prev_dist_p= curr_var;
                     end

             end

            end
    end

% Graph plotting starts here

%   clf
%   color=char('r','b','m','g','y','c','k');
%
%   for index=1:1:num_cls
%
    exp=strcat('plot(x(IDX==',num2str(index),',1),x(IDX==',num2str(index),',2),'",color(rem(index,7)),'.',"Marke
rSize",12)');
%   eval(exp);
%   exp='hold on';
%   eval(exp)
%   end
%
%   plot(C(:,1),C(:,2),'ko',...
%            'MarkerSize',12,'LineWidth',2)
%
%   exp='legend(';
%
%   for index=1:1:num_cls
%   exp=strcat(exp,"'Cluster ',num2str(index),"',');
%   end
%   exp=strcat(exp,"'Centroids","Location","NW")');
%   eval(exp);

%Graph plotting ends here


% 3D graph plotting starts here
tri = delaunay(x(:,1),x(:,2));
consumed = max_energy-power;
h = trisurf(tri,x(:,1),x(:,2),consumed);
```

```
%axis vis3d
x_max = max(x(:,1));
y_max = max(x(:,2));
p_max = max(consumed)+200;
axis([0 x_max 0 y_max 0 p_max]);
xlabel('Sensor X position')
ylabel('Sensor Y position')
zlabel('Energy Consumed (mJ)')
lighting phong
shading interp
colorbarEastOutside


% 3D graph plotting ends here
    A=c_head;   % contains the cluster head id's of each cluster
    B=IDX;      % contains the cluster id's of each sensor
    C=cl_count;  % contains the size of each cluster

end

function [IDX,C] = k_means(x,num_cls)
    [IDX,C] = k_means(x,num_cls)
end


function [IDX,C] = fuzzy(x,num_cls)
    [C,U]= fcm(x,num_cls);      % C is centers of each cluster
    IDX = zeros(size(x,1),1);

    rng(1);
    for i=1:1:size(x,1)
            temp_var = rand;
            curr_prob = 0;
      for j=1:1:num_cls
curr_prob = curr_prob + U(j,i);
        if temp_var<= curr_prob
          IDX(i,1)= j;
          break;
            end
       end
    end
end


function [IDX,C] = gauss_mix_model(x,num_cls)
    rng(1);
    obj = gmdistribution.fit(x,num_cls,'Regularize',0.01);
    IDX = cluster(obj,X);

    post_prob = posterior(obj,X);
    d = size(x,2);

    C=zeros(num_cls,d);
    for i=1:1:num_cls
            [maxi,index] = max(post_prob(:,i));
            for j=1:1:d
                    C(i,j) = x(index,j);
            end
    end

end
```

Explanation :-

Here is a step-by-step explanation of the code:

1. function [A,B,C] = clustering(x,scount,num_cls,power,max_energy)`. It takes five input arguments (`x`, `scount`, `num_cls`, `power`, `max_energy`) and returns three output variables (`A`, `B`, `C`).

2. The code defines a variable `Clustering_Method` to specify the clustering algorithm to be used. The value of this variable determines the algorithm chosen for clustering.

3. The code then checks the value of `ClusteringMethod` to select the appropriate clustering algorithm. If the value is 1 or 3, the K-means algorithm is used. If the value is 2 or 4, the Fuzzy C-means algorithm is used. The code uses the functions `kmeans` and `fuzzy` to perform the respective clustering algorithms.

4. After clustering the data using the chosen algorithm, the code calculates the distance from each sensor to the centroid of its cluster and determines the size of each cluster. This is done by iterating over each cluster and each sensor to count the number of sensors in each cluster and calculate the distance between each sensor and its cluster centroid.

5. If the clustering method involves power (methods 3 and 4), the code calculates additional information related to power. It calculates the maximum distance and maximum remaining power for each cluster.

6. The code then performs cluster head election. For each cluster, it iterates over the sensors in that cluster and selects the sensor with the minimum distance and power criteria as the cluster head.

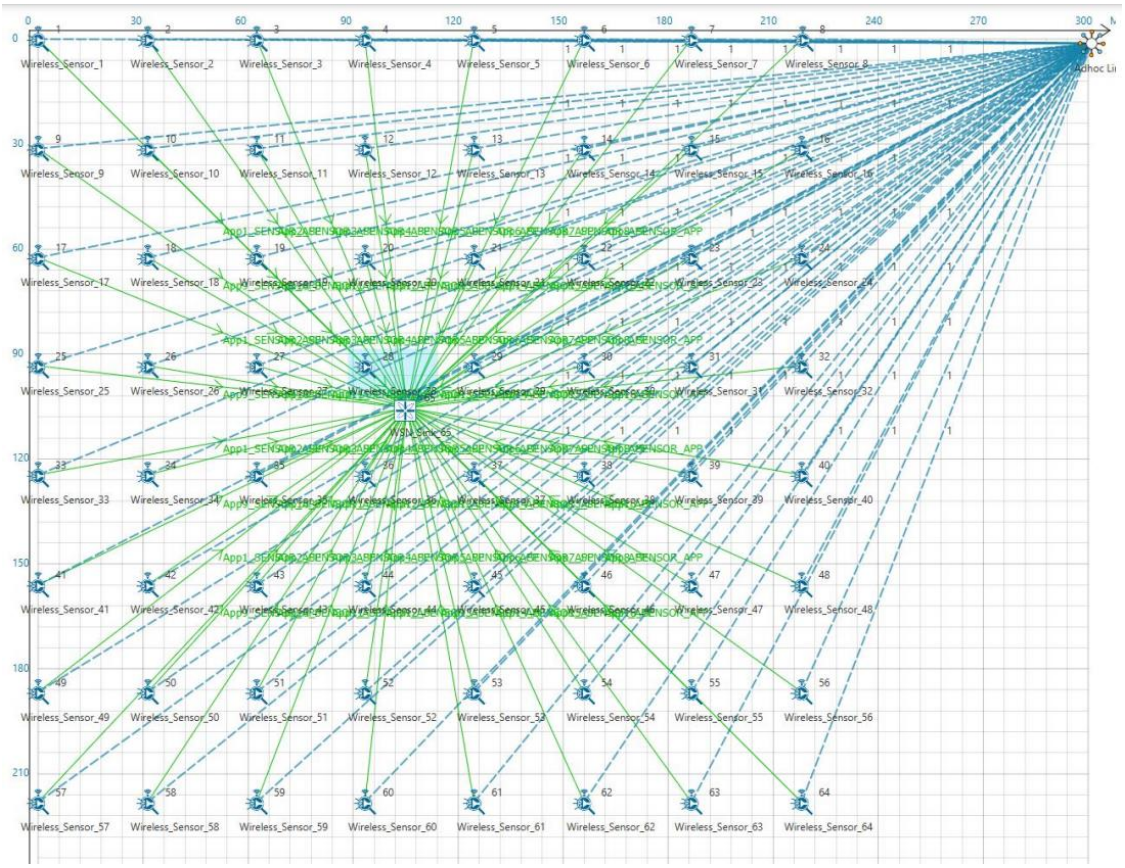
7. The code includes commented sections for graph plotting using MATLAB's plot functions. It plots the data points for each cluster, the centroids of the clusters, and adds a legend to identify the clusters.

8. The code includes 3D graph plotting using MATLAB's trisurf function. It plots the data points in a 3D space based on their X and Y positions and represents the energy consumed on the Z-axis.

9. Finally, the cluster head IDs, cluster IDs, and cluster sizes are assigned to the output variables (`A`, `B`, `C`) and returned by the function.

10. It's important to note that the code provided is not a complete implementation of the clustering algorithms. It includes the main steps for clustering and some partial calculations but lacks the complete implementation of the clustering algorithms.

7. Result :-



8. Conclusion :-

In conclusion, the provided code is a MATLAB function for performing clustering on a set of data points using different methods such as K-means and Fuzzy C-means. It includes steps for selecting a clustering method, calculating distances and cluster sizes, performing cluster head election, and includes optional code for graph plotting.