APPENDIX B: Pseudo-Code of Clustering Algorithms

The pseudo-code of the simulated algorithms in this dissertation is presented here for reference. The network simulation parameters are all same for existing and proposed algorithms. The algorithm specific parameters are mentioned here along with the pseudo-code.

Energy consumption model:

The transmission energy and reception energy models used are same as in LEACH.

\[
Transmission\ energy\ E_{Tx} = \begin{cases} 
    lE_{elec} + \epsilon_{fs} \ d^2, & d < d_{thres} \\
    lE_{elec} + \epsilon_{mp} \ d^4, & d \geq d_{thres} 
\end{cases} \quad (A.1)
\]

\[
Reception\ energy\ E_{Rx} = lE_{elec} \quad (A.2)
\]

\(E_{T_x}\) and \(E_{R_x}\) denote the transmission and reception energy in Equation (A.1) and (A.2). \(l\) denotes number of bits to be transmitted; \(E_{elec}\) is the electronics energy spent for node activities like transmission and reception. \(\epsilon_{fs}\) denotes energy dissipation in free space propagation and \(\epsilon_{mp}\) is the energy dissipation during multipath propagation. \(d\) represents the distance between two nodes and \(d_{thres}\) is the threshold distance to determine whether the transmission model is free space or multipath propagation model. All the simulation parameter values are mentioned in Table A.1.
Table A.1: Simulation Parameters and Values

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{elec}$</td>
<td>50nJ per bit</td>
</tr>
<tr>
<td>$\varepsilon_{fs}$</td>
<td>10pJ per bit</td>
</tr>
<tr>
<td>$\varepsilon_{mp}$</td>
<td>0.0013pJ per bit</td>
</tr>
<tr>
<td>$d_{thres} – distance$</td>
<td>87m</td>
</tr>
<tr>
<td>Initial energy per node</td>
<td>1J</td>
</tr>
<tr>
<td>Size of Data message</td>
<td>500 bytes</td>
</tr>
<tr>
<td>Size of Control message</td>
<td>25 bytes</td>
</tr>
<tr>
<td>Number of sensors</td>
<td>100</td>
</tr>
</tbody>
</table>
LEACH:

LEACH Algorithm:

1. \( N \) = Total number of nodes
2. \( p \) = desired percentage of CH
3. \( r \) = current round
4. \( \text{nodes} \) = set of alive nodes
5. for \( i=1:N \)
6. \( T(n) = \frac{p}{1-p^* (r \mod \frac{1}{p})} \)
7. \( x = \text{rand}(0,1) \)
8. if \( (x < T(n)) \)
9. \( \text{state}(i) = \text{CH} \)
10. \( \text{broadcasts CH_Election} \)
11. else
12. \( \text{State}(i) = \text{CM} \)
13. end
14. for \( i=1:N \)
15. if \( \text{(state}(j) = \text{CM}) \)
16. \( \text{send Join_Request to nearest CH node} \)
17. end
18. end

The desired percentage of Cluster head is set as 0.1 (i.e.) \( p=0.1 \) in all the simulations for LEACH.
LEACH – C:

**LEACH - C Algorithm:**

1. \( N \) = Total number of nodes
2. \( FCH \)=Array of cluster heads
3. \( FCM \)=Array of cluster members
4. \( p \) = desired percentage of CH
5. for \( i=1:N \)
6. Transmission to BS with residual energy, location details
7. end
8. Elects CHs based on \( p \) value with less intra-cluster distance
9. for \( j=1:N \)
10. Transmits the status of each node
11. if (CH==true)
12. informs about its CH election for current round
13. \( FCH=\text{node}(j) \)
14. else
15. informs the cluster membership details to the nodes
16. \( FCM=\text{node}(j) \)
17. end
18. end
ADRP:

ADRP Algorithm:
1. N = Total number of nodes
2. FCH=Array of cluster heads
3. NFCH=Array of next cluster heads
4. FCM=Array of cluster members
5. **Phase I** – Initial Phase
   - **Partition Stage**
     - for i=1:N
       - Transmission of residual energy and location details
     - end
     - Nodes with enough energy is allowed to participate in CH election
     - Elects CH with minimum CH and member node distance
     - Join elected CH to FCH
     - FCM=setdiff(nodes,FCH)
   - **Selection Stage**
     - NFCH=Next set of CHs are elected
   - **Advertisement Stage**
     - Current set of CH and Next CH set is broadcasted
     - Broadcast NFCH
6. **Phase II** – Cycle Phase
   - **Schedule Stage**:
     - for j = 1: size(FCH)
       - CH distributes TDMA schedule
     - end
   - **Transmission Stage**:
     - for l = 1:size(FCH)
       - for g= 1:length(Cluster)l
         - Data transmission from members to respective CH
       - end
     - end
   - **Re-cluster Stage**:
     - for p = 1:N
       - switch to the next CH node given by BS
     - end
EECABN:

**EECABN Algorithm:**

1. $N =$ Total number of nodes
2. $K =$ number of CH
3. $M =$ number of members
4. $RE =$ residual energy of nodes
5. $ND =$ neighbor node distance
6. $Dst\_to\_BS =$ Distance to BS
7. for $i=1:N$
8. Data transmission from sensor nodes to BS with RE and location details
9. compute weight ($RE$, $ND$, $Dst\_to\_BS$) of nodes
10. end
11. Nodes with higher weights elected as CH
CHEF:

**FLECH Algorithm:**

1. \( N = \text{Total number of nodes} \)
2. \( i = \text{sensor node} \)
3. \( \text{for } i=1:N \)
4. \( x=\text{rand}(0,1) \)
5. \( \text{if } (x<P_{\text{thres}}) \)
6. \( \text{state } (i)=\text{probationary CH} \)
7. \( \text{RE}(k)=\text{residual energy of the node.} \)
8. \( \text{LD}(k)=\text{Local Distance of the node within its } R_C \)
9. \( \text{chance } = \text{FuzzyChance}(\text{RE}(k), \text{LD}(k)) \)
10. \( \text{broadcasts } \text{tentative}_CH \)
11. \( \text{else} \)
12. \( \text{state } (i)=\text{CM} \)
13. \( \text{end} \)
14. \( \text{for } m=1:N \)
15. \( x(m)=\text{list of all neighbor } \text{tentative}_CH \)
16. \( \text{this}=\text{current sensor node’ } m’ \)
17. \( \text{if } (\text{state } (m)==\text{tentative}_CH) \)
18. \( \text{status}=1 \)
19. \( \text{for } t=1:\|x(m)\| \)
20. \( \text{if } (\text{this.chance}>\text{chance}(t)) \)
21. \( \text{continue} \)
22. \( \text{else} \)
23. \( \text{status}=0 \)
24. \( \text{break} \)
25. \( \text{end} \)
26. \( \text{end} \)
27. \( \text{if } (\text{status}==1) \)
28. \( \text{state } (m)=\text{Final CH} \)
29. \( \text{advertise } \text{Elected}_CH \)
30. \( \text{else} \)
31. \( \text{state } (m)=\text{CM} \)
32. \( \text{advertise } \text{Quit}_CH \)
33. \( \text{end} \)
34. \( \text{end} \)
35. \( \text{for } w=1:N \)
36. \( \text{if } (\text{state } (w)==\text{CM}) \)
37. \( \text{send } \text{Join}_\text{Req} \text{ to closest CH} \)
38. \( \text{end} \)
39. \( \text{exit} \)

\( P_{\text{thres}} \) in CHEF is set at 0.3 as specified in their work.
EAUCF:

**EAUCF Algorithm:**

1. \( N = \) Total number of nodes
2. \( i = \) sensor node
3. \textbf{for} \( i=1:N \)
4. \( x=\text{rand}(0,1) \)
5. \textbf{if} \( (x<P_{\text{thres}}) \)
6. \( i=\text{probationary CH} \)
7. \( \text{state}(i)=\text{tentative CH} \)
8. \( \text{RE}(k) = \) residual energy of the node.
9. \( \text{dist}_\text{to BS}(k) = \) distance to BS
10. competition radius = \( \text{EAUCF}(\text{RE}(k), \text{dist}_\text{to BS}(k)) \)
11. Broadcast \( \text{RE}(i) \) within competition radius
12. \( x=\) list of all \textit{tentative CH} within competition radius
13. \textbf{else}
14. \( i=\text{member} \)
15. \textbf{end}
16. \textbf{for} \( j=1:N \)
17. \textbf{if} \( (\text{state}(j)==\text{tentative CH}) \)
18. \( \text{status}=1 \)
19. \textbf{for} \( m=1:|x(j)| \)
20. \textbf{if} \( (\text{this.RE}>\text{RE}(m)) \)
21. \( \text{continue} \)
22. \textbf{else}
23. \( \text{status}=0 \)
24. \( \text{break} \)
25. \textbf{end}
26. \textbf{if} \( (\text{status}=1) \)
27. \( \text{state}(j)=\text{CH} \)
28. \( \text{Advertise Elected CH} \)
29. \textbf{else}
30. \( \text{state}(j)=\text{CM} \)
31. broadcasts \textit{Quit CH}
32. \textbf{end}
33. \textbf{end}
34. \textbf{for} \( w=1:N \)
35. \textbf{if} \( (\text{state}(w)==\text{CM}) \)
36. \( \text{send Join_Req to closest CH} \)
37. \textbf{end}
38. \textbf{exit}
EADC-FL:

EADC-FL Algorithm:

1. \( N \) = Total number of nodes
2. \( i \) = sensor node
3. \( nd \) = node degree
4. \( nc \) = node centrality
5. **Phase I – Information collection Phase**
   - For \( i=1:N \)
     - Broadcasts Node_msg with node id and remaining energy
   
   End
   - Calculates avg_rem_energy
6. **Phase II – Candidate CH competition Phase**
   - Wait for timer which is inversely proportional to remaining energy
   - After timer expiration issue CH_msg or Withdraw msg
7. **Phase III – Finalization Phase**
   - \( Z = \text{size(probationary CH)} \)
   - For \( k=1:Z \)
     - \( k.nd \) = number of neighbors
     - \( k.nc \) = node centrality
     - Cost = \( \text{EADC-FL}(k.nd, k.nc) \)
   
   End
   - For \( m=1:Z \)
     - Send tentative_CH to all neighbor probationary CH nodes
     - \( x \) = list of all tentative_CH from neighbor nodes
     - this = current sensor node’ m’
     - if (this.cost<cost(x))
       - advertise Elected_CH
     - else
     - broadcasts Quit_CH
   
   End
   End
8. **Phase IV – Cluster Formation Phase**
   - FCH = Elected CH nodes
   - FCM = list of non-CH nodes
   - for \( w=1:\text{size(FCM)} \)
   - send Join_Req to closest CH
   - End
   - EXIT
ECPF:

ECPF Algorithm:

1. \( N = \) Total number of nodes
2. \( i = \) sensor node
3. \( nc=\) node centrality
4. \( nd=\) number of neighbors
5. for \( i=1:N \)
   a. Wait for the timer inversely proportional to \( RE \)
   b. After timer expiration if no \( CH\_msg \), then issue \( CH\_msg \)
   c. Before timer expiration any \( CH\_msg \) broadcast then withdraw from competition
6. \( Z=\) size(probationary CH)
7. for \( k=1:Z \)
   a. \( k.\)node degree = number of neighbors
   b. \( k.\)node centrality = nodes centrality with neighbors
   c. Cost = \( \text{ECPF}(k.\text{nd}, k.\text{nc}) \)
8. end
9. for \( m=1:Z \)
10. Send \textit{tentative}_\textit{CH} to all neighbor probationary CH nodes
11. \( x=\) list of all \textit{tentative}_\textit{CH} from neighbor nodes
12. this=current sensor node’ \textit{m’}
13. if (this.Cost<Cost(x))
14. advertise \textit{Elected}_\textit{CH}
15. else
16. broadcasts \textit{Quit}_\textit{CH}
17. end
18. end
19. FCH=\textit{Elected} CH nodes
20. FCM=list of non-CH nodes
21. for \( w=1:\) size(FCM)
22. send Join_Req to closest CH
23. End
24. EXIT
GCA:

Fitness function of GCA is given below,

\[ \text{Fitness function (GCA) = } \omega \ast N_{CH} + (1 - \omega) \ast ND \]  \hspace{1cm} (A.3)

where \( \omega \) corresponds to weight value and it is set at 0.5 as mentioned in GCA, \( N_{CH} \) represents the number of CH and ND represents network distance (i.e.) sum of distance of the neighbor nodes to a particular node.

**GCA Algorithm:**

1. Generate Random initial population
2. Evaluate the initial solutions using fitness function
3. For 1: till the condition is true
   a. Apply Elitism selection operator
   b. Apply single point crossover, \( p_c = 0.6 \)
   c. Apply mutation with given probability, \( p_m = 0.03 \)
4. Update the population with new offspring
5. End
6. Select the best fit chromosome and form the cluster accordingly
EAERP:

**EAERP Algorithm:**

1. Generate Random initial population
2. Evaluate the initial solutions using fitness function
3. for 1: till the condition is true
   a. Apply Elitism selection operator
   b. Apply single point crossover, \( p_c = 0.6 \)
   c. Apply mutation with given probability, \( p_m = 0.03 \)
4. Update the population with new offspring
5. end
6. Select the best fit chromosome and form the cluster accordingly

Fitness function of GCA is given below,

\[
\Phi_{EAERP}(I^k) = \left( \sum_{i=1}^{nc} \sum_{sec} E_{Txs,ch_i} + E_{Rx} + E_{DA} \right) + \sum_{l=1}^{nc} E_{Txch_l,BS} \quad (A.4)
\]

where \( nc \) corresponds to number of clusters, \( E_{Rx} \) represents reception energy and \( E_{Tx} \) represents transmission energy.