A STUDY OF LOCALIZED ALGORITHM FOR SELF-ORGANIZED WIRELESS SENSOR NETWORK AND IT'S IMPLEMENTATION

Arnab Kundu

Asst. Professor, Department of E.C.E, B.I.E.T, Suri, Dist.-Birbhum, PIN-731101, W.B., INDIA, ***a.kunduwb@gmail.com***

Abstract

Communication between two nodes is a big issue in now days. To achieve that, wireless network plays a big role. With limited source of energy, memory & computation power wireless sensor networks can be composed by mass number of sensor nodes. Where the sensors' lifetime depend on energies. Like autonomous system in LAN, cluster can be defined for wireless sensor network, where cluster head plays the prime role for the energy conservation. So, the optimization of cluster head within cluster along with number of nodes is a big research issue. Here I have analyzed the advanced optimization algorithm for sensor network clustering theoretically. I have tried to test the proposed method as a clustering algorithm and compared it with other recent sensor network clustering algorithm named Algorithm for Cluster Establishment (ACE)

Index Terms: *ACE; Data sets; Localized algorithm; Migration; Spawning threshold function*

---***---

1. INTRODUCTION

The unique characteristics of wireless sensor networks and the nature of their applications motivate the development of new algorithmic solutions to provide services such as information processing, messages routing, fault-tolerance, localization, naming and addressing. Generally algorithms have been classified as either centralized or distributed. Centralized algorithms execute on a single node, which is impractical for large-scale highly constrained sensor networks. This approach would require all nodes to send their respective information (e.g., sensed data, or energy level) to a specified location (e.g., a sink node) and wait until this node executes the algorithm and sends back its output[1,6]. This approach is unsuitable for sensor networks because of its cost in terms of energy and delay. In addition, it suffers low fault-tolerance and low scalability and creates bottlenecks. In the case of distributed algorithms, different nodes participate in the computation which can reduce the execution time. To address this problem, a new class of algorithms called localized algorithms has been proposed for sensor networks. Localized algorithms are a special category of distributed algorithms that are designed specifically to enable efficient and reliable collaboration between sensor nodes by taking into account their specific constraints. For instance, a localized algorithm might limit the collaboration and information exchange to close neighbors of a node to save energy.[2,4,9,10]

2. (ACE): ALGORITHM FOR CLUSTER ESTABLISHMENT

Unlike other distributed clustering schemes, ACE employs an emergent algorithm. Emergent algorithms much like artificial neural networks evolve to optimal solution through a mix of local optimization steps. The main idea of ACE is to allow a node to assess its potential as a CH before becoming one and stepping down if it is not the best CH at the moment. The algorithm works in iterations that do not have to be synchronized at the individual nodes. Spawning new clusters and migration of existing ones are the two functional components of ACE. A node spawns of new cluster when it decides to become a CH. It broadcasts an invitation message to recruit its neighbors. Upon getting the invitation, a neighboring sensor joins the new cluster and becomes a follower of the new CH. At any moment, a node can be a follower of more than one cluster. However, the node can be a loyal follower, i.e. a member, of only a single cluster.[3,5]

2.1 Migration

Migration is a process in which the best candidate for being CH is selected. Each CH periodically checks the ability of its neighbors for being a CH and decides to step down if one of these neighbors has more followers than it does. A node that has the largest number of followers and the least overlap with existing clusters will be considered as the best candidate for CH.

2.2 Parameter Selection

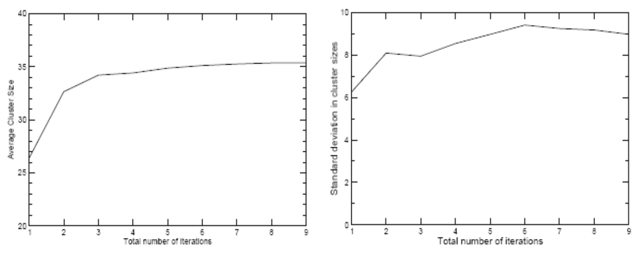
In the ACE protocol, an unclustered node will spawn a new cluster by declaring itself a cluster head whenever it finds that it can gain at least fmin loyal followers if it were to become a cluster head. The function fmin is called the spawning threshold function and is dependent on the time that has passed since the protocol was initiated for that node. In general, fmin should decrease as the algorithm proceeds. This causes fewer clusters to form near the beginning of the algorithm. Here, I used an exponentially decreasing function for fmin:

$$f_{min} = (e^{-k_1 \frac{t}{cI}} - k_2)d$$

In this formula, t is the time passed since the protocol began and cI is the duration of the protocol as described earlier. d is the estimated average degree (number of neighbours) of a node in the network. k1 and k2 are chosen constants that determine the shape of the exponential graph. In practice, it was empirically found that k1 = 2:3 and k2 = 0:1 have produced good results. In this case, fmin starts at 0:9d at the beginning of the protocol and reduces to 0 by the final iteration. This ensures that any node left unclustered at the end of the protocol will declare itself a cluster head.[7]

An alternative parameter setting is k1 = 2:3 as before, but setting k2 = 0. In this case the function starts near d when the protocol commences and reduces to 0:1d at the end of the protocol. Since 0:1d > 1 if d > 10, it is possible that there will be a small number of nodes that will not be within one hop radius of any cluster-head at the end of the algorithm. It can be observed in simulation that the number of nodes not within one-hop radius of a cluster-head is, on average, less than 4% of the total number of nodes in low node deployment densities, and around 2% for moderate to high node deployment densities (20 or more neighbors per node). These nodes that are not within one hop radius of any cluster-head can simply pick a clustered neighbor to act as their bridge to the cluster head, thus becoming two-hop followers (because they take 2 hops to communicate with the cluster-head, instead of the usual 1 hop).

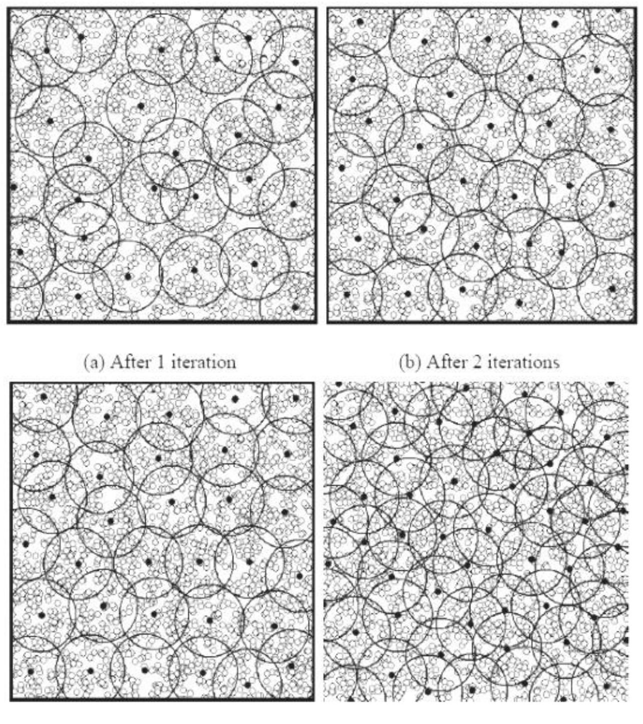
It remains to determine c, the number of iterations the algorithm should execute. Figure-1. reflects how the performance of ACE changes as it is given a longer number of iterations to operate. ACE was simulated in a 2D area with a uniform random distribution with an average deployment density d of 50 nodes per circle of one communication radius. Results for the simulation with k1 = 2:3 and k2 = 0:1 are shown (results for k2 = 0 have similar characteristics). It can be noted that increasing the number of iterations above 3 yielded only very slight improvements in average cluster size. In this simulation the total number of iterations did not significantly affect the standard deviation in cluster sizes, which was between 6 and 10 for all iterations > 1. [3,5]



(a) Average cluster size (b) Standard deviation of cluster size

Fig-1 Performance of ACE at various maximum iterations, d=50, k1=2.3, k2=0.1

Figure-2. illustrates the ACE algorithm operating in simulation (with k1 = 2:3 and k2 = 0). The little circles represent nodes. Cluster-heads are highlighted in black, and their range is indicated with a large black circle (nodes within the circle are in that cluster). The clusters migrate away from each other in successive iterations to produce a highly efficient cover of the area. Clusters tend to center over areas where nodes are dense. The clusters overlap minimally, and when they do overlap, they tend to overlap in areas where nodes are sparse. Figure-2d. provides a qualitative visual comparison of the Node ID algorithm with ACE. It can be observed that ACE provides a packing with significantly less cluster overlap than Node ID.[8]



(a) After 1 iteration (b) After 2 iterations

(c) After 3 iterations (d) *The Node ID algorithm (for comparison)*

Fig-2 The ACE algorithm (with k1=2.3 & k2=0)

2.3 ACE Algorithm

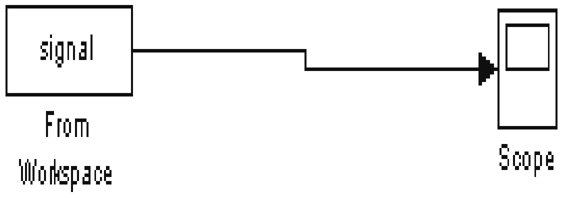
```
procedure SCALE_ONE_ITERATION()
  if $myTime > 3\times$ EXPECTED ITERATION LENGTH then
    if $myState$ = CLUSTER-HEAD then
      return DONE
    else if $myState$ = CLUSTERED then
      wait for my cluster-heads to terminate, then pick one as my cluster-head
      return DONE
    else if $myState$ = UNCLUSTERED then
      pick a random clustered node to act as my proxy after it terminates
      wait for it to terminate, then return DONE
    end if
  else if $myState$ = UNCLUSTERED
  and numLoyalFollowers() $\geq f_{min}(myTime)$ then
    $myClusterID \leftarrow$ generate_New_Random_ID()
    locally_broadcast(RECRUIT, $myID$, $myClusterID$)
  else if $myState$ = CLUSTER-HEAD then
    $bestLeader \leftarrow myID$
    $bestFollowerCount \leftarrow$ numLoyalFollowers
    for all $n$ where $n$ is a potential new cluster-head do
      $followerCount$ = Poll_For_Num_Loyal_Followers($n$, $myClusterID$)
      if $followerCount > bestFollowerCount$ then
        $bestLeader \leftarrow n$
        $bestFollowerCount \leftarrow followerCount$
      end if
    end for
    if $bestLeader$ is not $myID$ then
      send($bestLeader$, PROMOTE, $myClusterID$)
      wait for $bestLeader$ to broadcast it's RECRUIT message
      locally_broadcast(ABDICATE, $myID$, $myClusterID$)
    end if
  end if
end procedure
```

3. SIMULATION & DISCUSSION

To implement the above algorithm I have used the Application of Program named "MATLAB – 7" as front end application and "Microsoft Excel – 2000" as a Backend application or database. In this section I have shown the outputs of various type of Sensor nodes for variable Lat-Long. Then I have evaluated the function fmin in the following program.

3.1. Sensor Node Outputs

Data Name : elcentro_EW.dat
signal=load('elcentro_EW.dat');
From Simulink Library Browser :



Output: 1

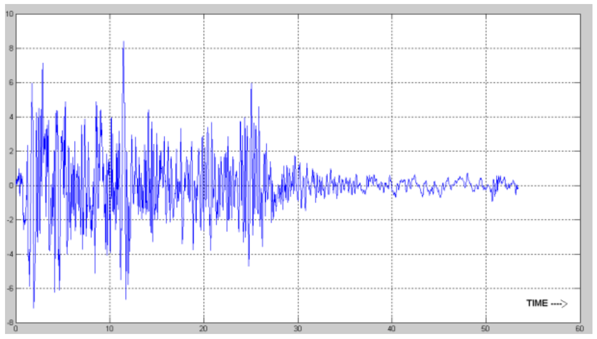
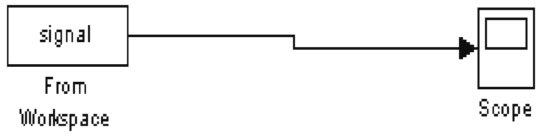


Fig-3 Vibration sensor output

Data Name : elcentro_NS.dat
signal=load('elcentro_ NS.dat');
From Simulink Library Browser :



Output: 2

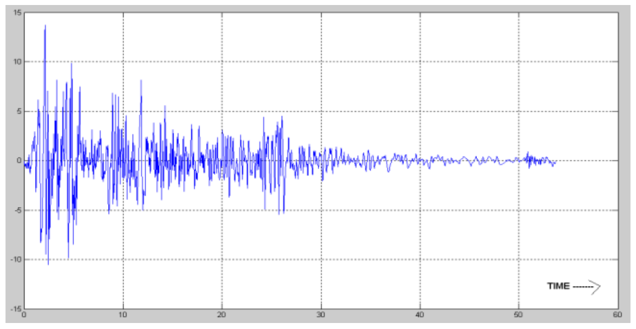
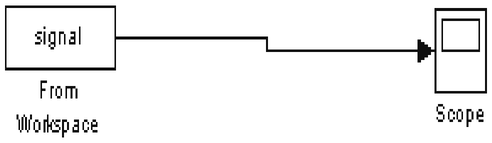


Fig-4 Vibration sensor output

Data Name: elcentro_EW.dat
signal=load(' elcentro_UP.dat');
From Simulink Library Browser :



Output: 3

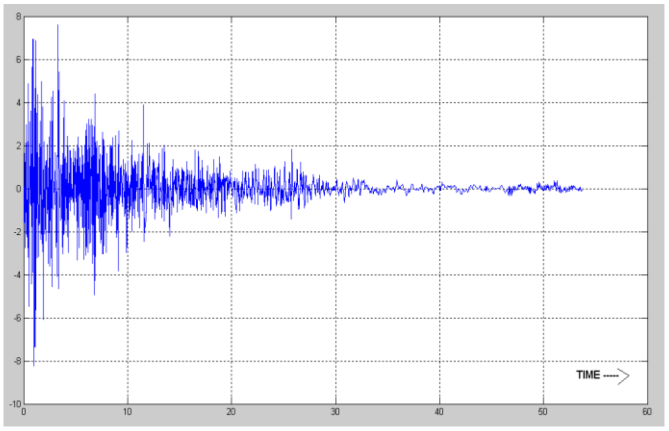


Fig-5 Vibration sensor output

Calculation of fmin

Considering, fmin = d
k1 = a
t/cI = b
k2 = c
d = h

Output: 1

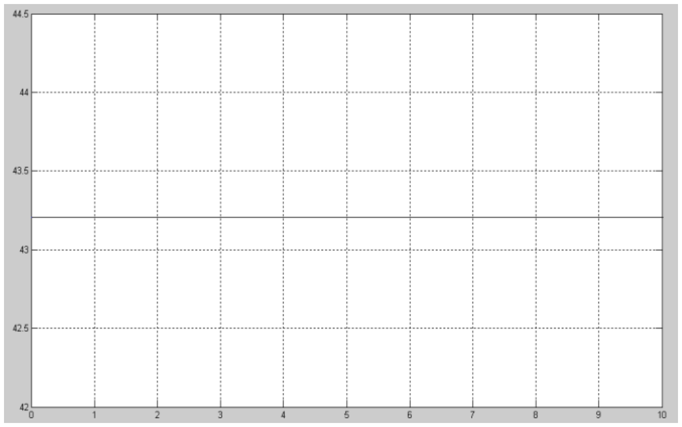
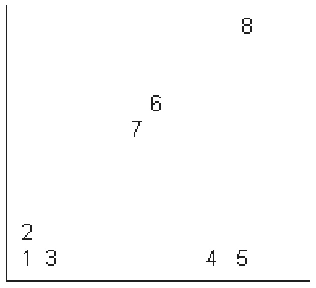


Fig-6 Performance of ACE at various maximum iterations, d=50, k1=2.3, k2=0.1

Each object is represented by an ordered n-tuple of its attributes. Objects with similar characteristics must have individual attributes which are close in value. Since distance is a measure of "closeness", it can be used in clustering. For each cluster, there will be a single point which is designated as a hub. The placement of the hub within the cluster is determined by the algorithm. I shall start with two-dimensional objects since they are the easiest to visualize. Suppose I have 8 points in the configuration below. Draw circles around the clusters that you see visually.

Table – 1

POINT	X	Y
1	1	1
2	1	2
3	2	1
4	8	1
5	9	1
6	6	6
7	5	5
8	9	9



Let's see what a computer model of clustering would give us. This algorithm assumes a minimum of 2 clusters and a maximum of p clusters where p is the number of objects. The algorithm is as follows:

1. From the set of objects (data points), choose any one to be the first hub.
2. Find the point which is farthest from this hub, and make it the second hub.
3. For each remaining object, find the distance from the object to each hub and assign the object to the hub to which it is closer.
4. To determine whether it is necessary to find a third hub, do the following.
    a. Find the distance between the two hubs and divide by two. Call this value R.
    b. Calculate the distance from each object to its hub. If any distance is greater than R, it is necessary to find a third hub.
5. To find the third hub, find the object which is farthest from its hub (1 or 2). So, if Point 8 belongs to Hub 1, look at the distance from Point 8 to Hub 1, but not Point 8 to Hub 2.
6. Then, calculate the distance from each point to the new hub to see if that distance is less than the point's distance to its current hub. If so, reassign the point to Hub 3.
7. To determine the stop value, R, find the average distance between the hubs and divide by two. Since there are three hubs, there will be three distances to average together (H1 - H2, H1 - H3, and H2 - H3).

8. Once again, see if any distance from a point to its hub exceeds R.
9. Continue this process until all points are within R units of their hub or until all points are themselves hubs.

Table – 2: data.dat (Data File)

	1	2
1	4	5
2	1	7
3	1	2
4	2	8
5	8	1
6	7	8
7	8	7
8	5	4

Output: 1.

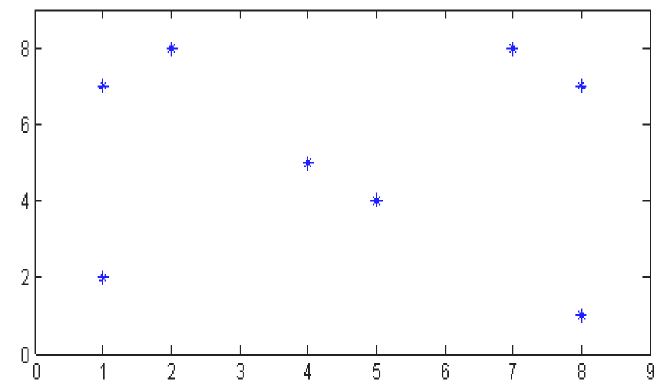


Fig-7 Graphical plotting of datasheets (Top View)

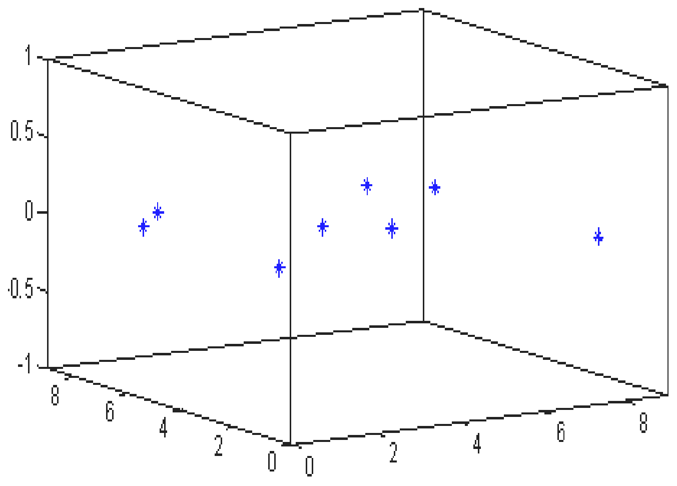


Fig-8 Graphical plotting of datasheets (3-D View)

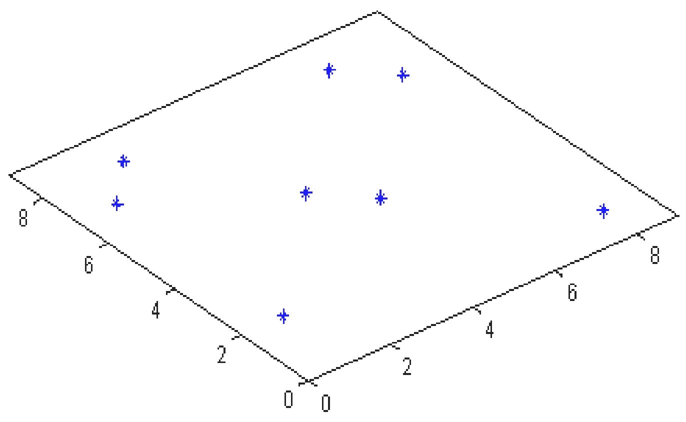


Fig-9 Graphical plotting of datasheets (Side top View)

Output: 2.

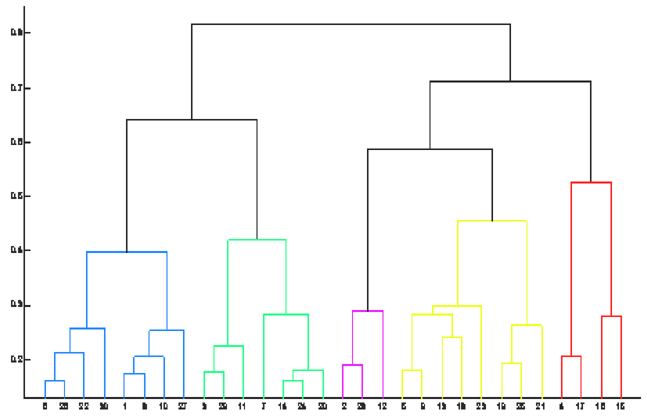


Fig-10 Clustering of Data Sets

Output: 3.

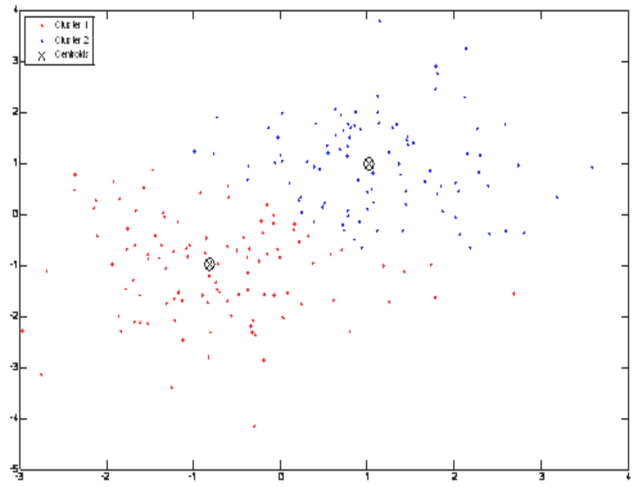


Fig-11 Clustering process

CONCLUSIONS

Through the above theoretical analysis and simulated outputs I am trying to solve the network link between two Nodes of a cluster as well as measure the Stability Criteria of those (A.C.E., S.O.S., Optimization Algorithm) Clustering Algorithms. Clustering Algorithms are very important at present. There are huge scope of work. The basic concepts needed to understand the functionality of the algorithms. This paper was meant as an introduction to the algorithms and also the purpose was to create an efficient way to display the clustering of individual data elements. If something remains to be done a more throughout analysis of the behavior of the algorithms should be made. Specially cases where the samples are badly balanced or in some particular order should be generated and analyzed.

ACKNOWLEDGEMENTS

The authors would like to thank the authorities of Birbhum Institute of Engineering & Technology & JIS College of Engineering for providing every kind of supports and encouragement during the working process.

REFERENCES

[1]. Alan D. Amis, Ravi Prakash, Thai H.P. Vuong, and Dung T. Huynh. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. In Proceedings of IEEE INFOCOM 2000.

[2]. D.J. Baker, A. Ephremides, and J.A. Flynn. The Design and Simulation of a Mobile Radio Network with Distributed Control. IEEE Journal on Selected Areas in Communication, 2(1):226–237, January 1984.

[3]. Suman Banerjee and Samir Khuller. A Clustering Scheme for Hierarchical Control in Wireless Networks. In Proceedings of IEEE INFOCOM 2001, April 2001.

[4]. Stefano Basagni. Distributed Clustering for Ad Hoc Networks. In Proceedings of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN), June 1999.

[5]. A Survey of Clustering Algorithms for WirelessSensor NetworksD. J. Dechene, A. El Jardali, M. Luccini, and A. Sauer.Department of Electrical and Computer EngineeringThe University Of Western OntarioLondon, Ontario, Canada

[6]. M. Gerla, T.J. Kwon, and G. Pei. On Demand Routing in Large Ad Hoc Wireless Networks with Passive Clustering. In Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2000), September 2000.

[7]. Rajesh Krishnan, Ram Ramanathan, and Martha Steenstrup. Optimization Algorithms for Large Self-Structuring Networks. In IEEE INFOCOM '99, 1999.

[8]. Chunhung Richard Lin and Mario Gerla. Adaptive Clustering for MobileWireless Networks. IEEE Journal of Selected Areas in Communications, 1997.

[9]. Clustering Algorithms: Basics and Visualization – By, -Jukka Kainulainen

[10]. Glenn Fung: "A Comprehensive Overview of Basic Clustering Algorithms", June 22, 2001

BIOGRAPHIES

Arnab Kundu, born in India, obtained his M.Tech degree from JIS College of Engineering, Kalyani, West Bengal. Now he is Asst. Professor of E.C.E. Department in Birbhum Institute of Engineering & Technology, P.O.:Suri, Dist.:Birbhum, PIN:731101, West Bengal, India. He is Life member of Indian Society for Technical Education, Indian Society of Remote Sensing, International Association of Engineers, The Institution of Electronics & Telecommunication Engineers. email: a.kunduwb@gmail.com