Dynamic Algorithms for Efficient and Robust Clustering in Ad hoc Networks

by Kyriakos Manousakis

TR 2004-25

Dynamic Algorithms for Efficient and Robust Clustering in Ad hoc Networks

Kyriakos Manousakis and John S. Baras
Institute for Systems Research and
Electrical and Computer Engineering
University of Maryland, College Park
College Park, MD, 20742

Abstract

Ad hoc networks are very important for scenarios where there is not fixed network infrastructure. These scenarios may appear both in the military and the commercial world. Even though there is much advancement in the area of these networks, the main drawback is that ad hoc networks do not scale well because the existing protocols (e.g., MAC, routing, security) cannot tolerate the dynamics of these networks when their size becomes larger than few decades of nodes(i.e. *O(10)*). A remedy to this problem is to apply these protocols in hierarchical manner. The hierarchy generation in these dynamic environments can be advantageous since the numerous topological changes can be tolerated easier and the various protocols can perform better when dealing with smaller groups of nodes. This division greatly reduces overall overhead (e.g., routing overhead with n nodes goes from $O(n^2)$ to $O(n\log n)$) and allows protocols to be tuned to more homogenous conditions [1].On the other hand, hierarchy has to be generated carefully in order to be beneficial for the network otherwise it may harm it, because of the imposed maintenance overhead. Towards this objective we have to take into consideration the network environment and design appropriately the hierarchy generation algorithms. The weakness of the existing network clustering algorithms is that they do not take into consideration the dynamics of the network environment, so in cases of increased mobility their overhead may deteriorate network performance instead of improving it. In this paper we present a new class of clustering algorithms, which includes both a centralized and a distributed algorithm. The basic characteristic of these algorithms is that they take into consideration the network dynamics for the generation of robust and efficient clusters. The centralized algorithm generates optimal clusters but can be applied only in slow mobile networks. The distributed algorithm can be applied in highly mobile networks and even though the quality of clusters is lower than the centralized generated ones, the algorithm still presents better scalability and robustness characteristics from the existing distributed algorithms.

1 Introduction

The dynamic nature of ad hoc networks and the lack of fixed network infrastructure are the main characteristics that prevent these networks from scaling to a large number of nodes. Although, significant progress has been accomplished for the advancement of ad hoc technology through the design of numerous protocols (e.g. MAC, routing, security), their poor scalability characteristics when the networks become mobile, is the limitation of these protocols. The existing ad hoc protocols have been incapable to cope with the dynamics of the network environment, especially when the network size is in the order of few decades of nodes (*O(10)*). The performance of these protocols significantly decreases with respect to the network topology changes that happen due to the instability of the links and the failure of participating nodes. Even though new protocols are designed the scalability problem persists.

The scalability issues of the existing ad hoc protocols and the inherent difficulty of the protocols to handle the dynamics of the network suggest that we cannot think of a scalable flat mobile ad hoc network (MANET). The dynamic generation of a hierarchy can remedy the problem, because the protocols will only have to scale to a limited number of nodes, as opposed to the entire network. To provide this hierarchy many dynamic clustering algorithms, mainly based on local distributed approaches, have been proposed in the literature [2][3][4][5]. Their drawback, however, is that they do not take into consideration the overall network environment. Indeed, in many cases, these algorithms harm network performance instead of improving it because of the re-clustering overhead they impose in a dynamic network. Toward this direction we propose a new class of hierarchy generation algorithms. The main objective of these algorithms is the generation of robust and efficient clusters, such that the reclustering overhead is minimized or eliminated. The generation of such hierarchy improves the network performance, due to the fact that the networking protocols are applied in hierarchical manner and the overhead imposed from the clustering algorithms is minimized.

Consider the following example that proves the importance of our approach. Assume the following network environment, where the nodes 1-7 are static, so there is no variation of their initial position during the lifetime of the network (i.e., these nodes can be sensor nodes). The nodes 8-11 are mobile nodes, but they are moving as a group so they are relatively static (i.e., group of soldiers). The former group of nodes follows a cyclic trajectory around the static nodes.

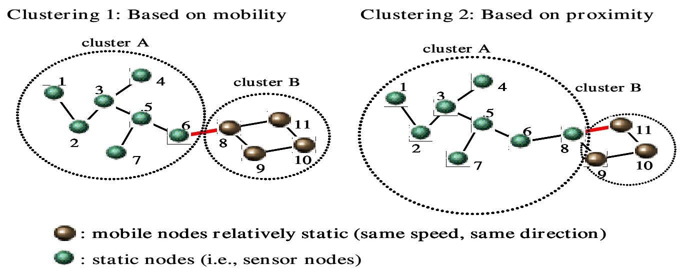

Clustering 1: Based on mobility
Clustering 2: Based on proximity
cluster A
cluster B
cluster A
cluster B
: mobile nodes relatively static (same speed, same direction)
: static nodes (i.e., sensor nodes)

Figure. 1. Dynamic Clustering Motivation Example (mobility vs. proximity)

In this case if we attempt to cluster based on the proximity of the nodes or utilize the traditional methods of clustering (lower ID, highest degree) [2][3] then we may end up with frequent re-clustering. The response of clustering algorithms to the topology changes is to re-cluster the network so as to maintain the clusters consistent to the principals of the specific algorithm. If we cluster with respect to the mobility of the nodes then re-clustering is not required, because, even though the positions of the nodes change, their mobility characteristics are very similar, and the connectivity between them remains almost unaffected. By using the mobility characteristics of the nodes as criterion for hierarchy generation we can achieve:

- More stable and robust clusters
- Minimize re-clustering/maintenance overhead

The proposed clustering algorithms take into account the mobility characteristics of the nodes. The nodes with similar mobility characteristics must be clustered together, so that they can remain in the same cluster for long periods of time before re-clustering occurs, which will penalize the network with extra overhead. In this paper we present two clustering algorithms, a centralized and a distributed. The former approach appears counter-intuitive for ad hoc networks (e.g. global network information required, centralized decisions). In particular, we study a centralized optimization algorithm based on Simulated Annealing [6] with a set of cost functions that are selected appropriately based on the network environment. Although the centralized global optimization provides significant benefits (i.e. obtains the most optimal clustering map), cannot deal with fast changing networks. The application of this algorithm is limited to scenarios where the network topology changes slowly, so that the network dynamics can be captured. Because of the optimality of the generated clustering map, the centralized algorithm can be utilized as a point of reference for the corresponding clustering maps that are generated from the distributed algorithm we propose.

On the other hand, the proposed distributed algorithm attempts to generate as robust clustering maps as the centralized clustering algorithm by applying similar metrics but in a localized manner. The algorithm requires one-hop information and presents $O(n)$ communication complexity in a network of n nodes. The generated clusters are more robust compared to the existing distributed clustering algorithms. Also, the algorithm presents very promising performance characteristics even in cases of large and highly mobile networks, where the existing distributed algorithms fail. The ability of the algorithm to deal with the dynamics of the network emerges from the fact that the hierarchy is generated with respect to these dynamics (e.g. mobility characteristics of the nodes).

In next section we define the mobility metrics we apply for the generation of robust clustering maps. In section 3 we present the centralized algorithm along with the corresponding mobility cost functions we propose. The presentation of the distributed algorithm follows in section 4. Section 5 deals with the performance evaluation of the proposed algorithms and their comparison to the existing clustering algorithms. We conclude this paper in section 6.

2 Mobility Metrics

Our main objective is to cluster together the network nodes such that the need for re-clustering due to the topology changes is minimized or even eliminated. In order to achieve this we have to take into consideration the dynamics of the network environment. If we assume that the participating nodes do not fail then the topology changes happen only because of the nodes mobility. By identifying the characteristics of nodes movement and cluster them accordingly then the generated clusters can be more stable in terms of their membership compared to other clustering schemes, which are based on characteristics uncorrelated to the mobility of the nodes. If we cluster together nodes that present similar mobility patterns then these nodes are expected to remain connected for a longer time periods compared to the case where we cluster together nodes with unrelated mobility characteristics. Based on the latter observation the membership information of the generated clusters does not have to change frequently, so the maintenance overhead will be minimized and the network will take full advantage of the generated hierarchy.

By identifying the mobility of the nodes as the source of the network topology changes, we have to define the metrics that characterize it. These metrics will be the building blocks of the cost functions. We define two classes of metrics. The first class has to do with the individual mobility of the nodes and the second class that has to do with their relative mobility characteristics. Even though the second class of metrics is more interesting, the first class of metrics is what the nodes will be able to measure in real time from the network. The relative mobility metrics can be extracted from the measured values of the individual mobility characteristics of the nodes.

The metrics that constitute the first class are:

- Direction (θ_i):The direction of a node *i* is described from the angle that is defined counter-clockwise from the straight line that is defined from two consecutive points on the trajectory of the node and the straight line parallel to the positive x-axis ($\theta = 0^o$). In real world scenarios a node can estimate its direction of movement utilizing various tools, for example a carry-on GPS device.

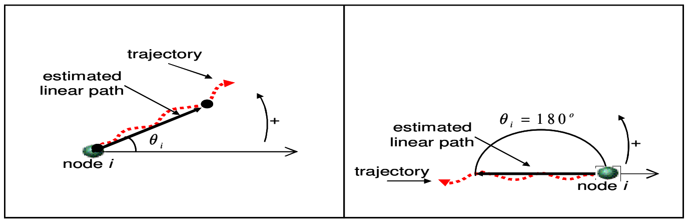


Figure 2. Node Direction: Computation Approach

- Speed (S_i): The speed of a node *i* is a measure of the rate of motion of this node. Defines the magnitude of the distance that is covered in a unit of time. In real world scenarios a node *i* can estimate the value of its speed utilizing various tools, for a example a carry-on GPS device.

If we know the values of the basic mobility metrics, then we can obtain the values of the relative metrics, which are important for the clustering of nodes. The reason that makes the latter metrics more important is that the connection stability of two nodes can be described better from their relative movement (speed and direction). For the same reason we prefer to cluster based on the relative mobility characteristics of the nodes rather on the individual ones. The second class of metrics includes the following:

- Relative Direction ($\theta_{r_{ij}}$): The relative direction of two nodes (*node$_i$*, *node$_j$*) is defined with respect to their individual directions of movement. If *node$_i$* is moving with direction θ_i and *node$_j$* with direction θ_j then their relative direction $\theta_{r_{ij}}$ is defines as:

$$\theta_{r_{i,j}} = \min\left(\left|\theta_i - \theta_j\right|, 360 - \left|\theta_i - \theta_j\right|\right), \tag{1}$$

where,

$$\theta_i, \theta_j \in \left[0^o, 360^o\right),\ \theta_{r_{i,j}} \in \left[0^o, 180^o\right]$$

A graphical example demonstrating the computation of the relative direction of two nodes follows:

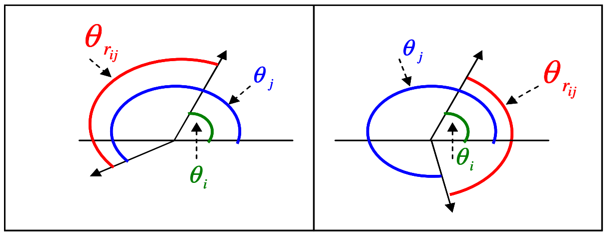


Figure 3. Relative Direction: Computation Approach

- Relative Velocity ($U_{r_{ij}}$): The relative velocity of two nodes is the velocity with which a node approaches or recedes from another node, whether both are moving or only one. The mathematical definition of this metric is:

$$U_{r_{i,j}} = \sqrt{U_{Y_{i,j}}^2 + U_{Y_{i,j}}^2}\,, \tag{2}$$

where,

$$U_{X_{i,j}} = S_X \cos\theta_i - S_X \cos\theta_j$$
$$U_{Y_{i,j}} = S_Y \sin\theta_i - S_Y \sin\theta_j$$

- Link Expiration Time (LET_{ij}): The Link Expiration Time is defined as the estimated lifetime of the link that connects two nodes, and is computed with respect to the mobility characteristics of the corresponding nodes (e.g., direction, speed). The mathematical definition of this metric is:

$$LET(j,k) = D_{t_i(j\leftrightarrow k)} = \begin{cases} \dfrac{-(ab+cd)+\sqrt{(a^2+b^2)r^2-(ad-bc)^2}}{a^2+c^2} & \text{,nodes } j,k \text{ are in range} \\ 0 & \text{,nodes } j,k \text{ are not in range} \\ \infty & \text{,nodes } j,k \text{ are relatively static} \end{cases} \tag{3}$$

where,

$$a = u_j \cos\theta_j - u_k \cos\theta_k$$
$$b = x_j - x_k$$
$$c = u_j \sin\theta_j - u_k \sin\theta_k$$
$$d = y_j - y_k$$
$$r = TxRange_{j,k} \quad (TxRange \text{ in this case is assumed the same for every node})$$

The second class of mobility metrics deals with the relative mobility characteristics of the nodes. By clustering together pairs of nodes that present similarities on the values of these metrics, we expect that the generated clusters will have longer life span and less membership changes compared to the existing algorithms, which utilize metrics unrelated to the dynamics of the network (i.e., number of hops).

3 Centralized Algorithm

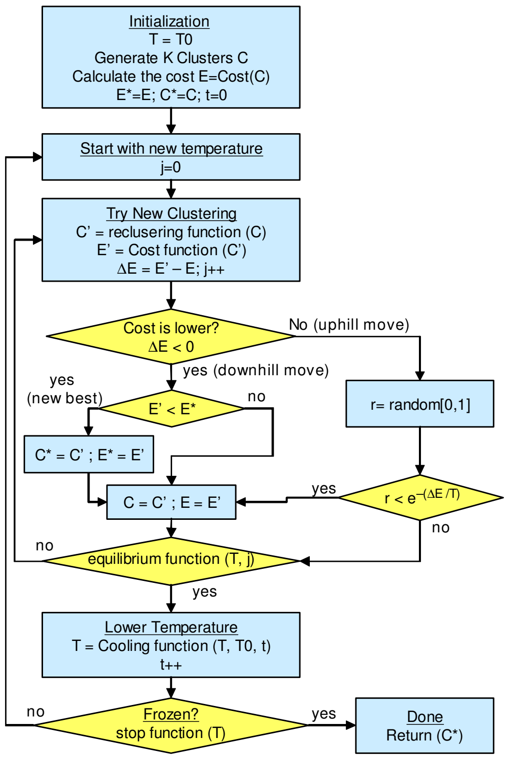
In this section we present the centralized approach for the dynamic clustering of the network. The core algorithm of this approach is the Simulated Annealing (SA) algorithm. SA achieves global optimization but it requires global information about the network and its original unmodified form presents slow convergence characteristics. Here we present the characteristics of the algorithm and the corresponding mobility cost functions we applied to generate robust clustering maps.

3.1 Simulated Annealing

Simulated annealing (SA) has been widely used for tackling different combinatorial optimization problems [7]. The process of obtaining the optimum configuration is similar to that followed in a physical annealing schedule. In SA, however, the temperature is merely used as a control parameter and does not have any physical meaning.

Figure 1 highlights the general steps in the algorithm. The objective of the algorithm is to obtain the K cluster network partition configuration, C^*, that optimizes a particular cost function. The process starts with an initial temperature value, T_0, which is iteratively decreased by the cooling function until the system is frozen (as decided by the stop function).

For each temperature, the SA algorithm takes the current champion configuration C^* and applies the recursive function to obtain a new configuration C' and evaluates its cost, E'. If E' is lower than the cost of the current E^*, C' and E' replace C^* and E^*. Also, SA randomly accepts a new configuration C' even though E' is greater than E^* to avoid local minima. In the latter case C' and E' replace C^* and E^* *respectively*. One of the key characteristics of simulated annealing is that it allows uphill moves at any time and relies heavily on randomization [6]. The higher the temperature, the higher the probability of accepting a configuration that worsens E^* instead of improving it. Indeed if the temperature is sufficiently high, SA will simply take a random walk around the solution space. The lower the temperature, the fewer the algorithm accepts worse configurations.



Inputs	Examples
Equilibrium function	Constant (j = 5000); "Stop repeats" (function of
Cost function	$\sum_{i=1}^{K} Diameter\ (C_i)$
Cooling function	Geometric or logarithmic
Stop function	Minimum temperature (e.g., T=0.1); "Stop repeat" criteria
Reclustering function	Random move of one node
T0	Initial Temperature
K	Number of clusters

Variable	Definition
T	Current Temperature
C	Current Cluser map (e.g., {1,2},{3,6},{4,5})
C'	New cluser map to test (eg. {1,2},{3,5,6},{4})
C*	Champion cluster map (eg. {1,6},{3,5},{2,4})
E	Current cost
E'	Cost of new cluster map
E*	Champion cost
j	Inner loop counter
t	Outer loop counter

Figure 4: Simulated Annealing algorithm for network partitioning

The number of iterations required to reach equilibrium are defined by the equilibrium function. The function can be a simple constant (e.g., 100) or a function of the temperature and other parameters specific to the optimization problem, such as number of nodes in the network.

In order to complete the description of the proposed dynamic clustering algorithm we will present the cost functions we suggest so as to succeed in meeting our objective, which is to minimize the imposed clustering maintenance overhead by generating stable clusters. The next section discusses the proposed mobility cost functions.

3.2 Cost Functions

The metrics we presented in section 3 are the cornerstones of the cost functions we propose. The careful definition of the cost functions is required for the fulfillment of our objectives. These cost functions will be optimized from the SA algorithm, subject to the constraint of generating topological clusters.

Definition (Topological Cluster): A cluster consisting of the set $\mathbb{S}$ of nodes is called topological if $\forall node_i, node_j \in \mathbb{S}$ and $i \neq j$, there is always a path P_{ij} from $node_i$ to $node_j$ s.t $\forall node_k \notin \mathbb{S}$ holds that $node_k \notin P_{ij}$. All the members of a cluster can communicate between them without the need to use inter-cluster links, which are links that involve non-member nodes.

The result of the optimization process will be a feasible clustering decision that is optimal subject to the applied cost function. The mobility cost functions we define are listed below:

- Cost Functions based on Relative Direction

  We preset cost functions that involve only the relative direction metric. Our objective is to generate topological clusters, which consist of nodes moving along the same direction. The two cost functions we propose are:

$$J(C) = \min\left[\sum_{z=1}^{K}\left(\sum_{i,j=1}^{|C_z|}\theta_{r_{i,j}}\right)^2\right] \tag{4}$$

$$J(C) = \min\left[\sum_{z=1}^{K} Var\left(\theta_{r^z_{1,2}}, \theta_{r^z_{1,3}}, \ldots, \theta_{r^z_{|C_z-1|,|C_z-1|}}, \theta_{r^z_{|C_z-1|,|C_z|}}\right)\right] \tag{5}$$

  These cost functions do not take into consideration the speed of the nodes so it is expected they will not be effective in cases where the nodes present similar directions but differ in their speed.

- Cost Functions based on Relative Velocity

  By utilizing cost functions that involve only the metric of relative velocity of the nodes, we want to cluster together nodes that are moving with similar velocities. The lower the relative velocity of a pair of connected nodes, the larger the possibility of clustering these nodes together subject to the constraint of constructing topological clusters. The corresponding cost function we applied is:

$$J(C) = \min\left(\sum_{z=1}^{K}\left[\sum_{i,j=1}^{|C_z|} U^2_{r^z_{i,j}}\right]^2\right) \tag{6}$$

  Even though the computation of relative velocity involves the direction of movement of the corresponding nodes, the value of relative velocity is a magnitude that does not describe explicitly the relative direction of movement. As the results show (6) behaves better than (4) and (5) but there is still room for improvement.

- Cost Functions based on Link Expiration Time

  The Link Expiration Time is another metric that is related to the relative mobility characteristics of the nodes. The following cost function is defined solely on this metric.

$$J(C) = \min\left[-\sum_{z=1}^{K} \left(\sum_{i,j=1}^{|C_z|} I_z\left(LET_{ij}\right) \right)^2 \right] \tag{7}$$

Cost function (7) indirectly is related to the speed and direction of the nodes. Also, it requires the coordinates of their location, which is information, not required for the computation of the other metrics. Even though (7) requires more information to be computed, there is still space for improvement in identifying the various mobility groups.

- Cost Functions based on Combination of Metrics

  The above cost functions are based solely on a single mobility metric. The weakness of these cost functions is when nodes that have different mobility characteristics are assigned in the same cluster because present similarity only on the mobility metric on which the corresponding cost function is based upon. To correct and improve the weakness of the previous cost functions we propose (8), which combines the relative direction and relative velocity.

$$J(C) = \min\left(\sum_{z=1}^{K} \left[\sum_{i,j=1}^{|C_z|} \frac{U_{r_{i,j}^{z}}}{2 * \max(S)} + \sum_{i,j=1}^{|C_z|} \frac{\theta_{r_{i,j}}}{180} \right]^2 \right) \tag{8}$$

  We expect that even in cases where the nodes present similarities in any of the two metrics but differ to the other one, they still, will be assigned in separate clusters.

4 Dynamic Distributed Algorithm – DDC

Although, the combination of SA algorithm and the proposed cost functions will provide us with optimal clustering decisions, the weakness of this approach is the centralized nature of SA. Due to the fact that SA requires global network information, processing time and distribution of the clustering decisions to the nodes, it makes this method unfavorable for large and highly mobile networks. In environments where the SA fails we cannot do better than applying a distributed heuristic. In this section we propose a distributed algorithm, which was designed to resemble the functionality of the centralized algorithm, by generating clusters with respect to the mobility metrics we presented in section 3.

The DDC algorithm compared to other distributed clustering algorithms, does not focus on selecting cluster heads and assigning nodes to these cluster heads based on the hop count [2][3][4][5]. The main objective is to group together nodes that present similar mobility characteristics, so that the generated clusters are robust to the topology changes. The goal of DDC algorithm is the same as of the SA algorithm, and it is to reduce the maintenance overhead, which is imposed to the network due to membership changes. Even though the generated clusters are not expected to be of the same quality compared to the robust clustering decisions of SA, but still the DDC algorithm can meet its objectives, as we show later, when we evaluate the performance of DDC.

4.1 Algorithmic Description of DDC

The Dynamic Distributed Clustering algorithm we propose is based on 1-hop information exchange. The Information is related to the mobility metrics we introduced in section 3. We assume that each node can obtain information about its speed, direction and position (e.g., only in the case where the metric of interest is the Link Expiration Time (LET)).

Apart from the 1-hop neighbor discovery the algorithm has 3 distinct phases. After the completion of these phases each node belongs to a cluster, where all the members present similarity with respect to the metric of interest. All three phases involve information exchange between 1-hop neighbors or a subset of them. Namely, the three phases are:

- Phase I – Neighbor Selection
  In this phase each node broadcasts its ID and its metric value to its 1-hop neighbors. The metric value is related to the metric of interest based on which the clusters will be formed. The metric can be related to the velocity, the direction or the position of the node or even to all or a subset of these values. All the nodes collect from their 1-hop neighbors these values and based on these values they select the

neighbor that best matches the objectives of the algorithm. For example someone would like to select the neighbor that moves to a direction similar to its own direction.

- Phase II – *InfoExchange* List Composition
  After the nodes have decided on which is the most appropriate neighbor to form a cluster with, they inform the selected node for this decision. The recipient nodes collect the related messages and record the IDs of the nodes who have selected them. After the completion of this process, each of the nodes generates the *InfoExchange* list, which is composed of the node IDs collected in Phase II, the ID of the selected neighbor and their own ID. The *InfoExchange* list is sorted in ascending order with respect to the node IDs.

- Phase III – Cluster Formation
  In this phase the nodes communicate based on the *InfoExchange* list from Phase II, in order to decide on a cluster. Each node communicates only with the nodes in its *InfoExchange* list in order to decide on the cluster that matches better its mobility characteristics, with respect to the metric of interest. The communication synchronization among the nodes is based on the node IDs (e.g., for that reason the *InfoExchange* list is sorted). Each node selects a cluster to join and transmits this decision after all nodes with lower IDs in its *InfoExchange* list have decided and transmitted their clustering decisions.

After the overview of the various phases of the DDC algorithm, its algorithmic steps follow in more detail:

```
Step 0: 1-hop NEIGHBOR DISCOVERY

Step I: (1-hop Communication)
      Send to every 1-hop neighbor a TYPE I message:
```

myID	MetricValue

```
Step II: (Processing)
       Collect the (TYPE I) messages from every 1-hop neighbor
         Based on the metric value related to every neighbor select the most appropriate1-hop neighbor
         with neighborID.

Step III: (Communication)
        Send to neighborID a TYPEII to inform it for its selection:
```

myID	neighborID

```
Step IV: (Processing)
        Collect all messages of TYPEII
        Generate the SelectedFromList which contains the neighborIDs that have selected myID as the
           preferred neighbor to be clustered with
        Generate the InfoExchangeList:
              $InfoExchangeList = SelectedFromList \cup neighborID \cup myID$
        Sort in ascending order the InfoExchangeList subject to nodeID

Step V: (Communication Part)
       IF $myID = Head(InfoExchangeList)$ then
          {
          $myCID = CID$
          Send to every node with $nodeID \in InfoExchangeList$ a TYPEIII message:
```

myID	CID=myID

```
          }
       ELSE {
```

```
    Until the reception of a TYPEIII message from the nodes with:
            $nodeID \in InfoExchangeList \wedge nodeID < myID$
    {
        Upon the reception of TYPEIII message {
            IF $myCID = \varnothing$ then
                        $myCID = CID$
            ELSE
              If $myCID > CID$ then myCID=CID
        }

        My turn to transmit:
            send $myCID$ to every node with $nodeID \in InfoExchangeList$

        Until the reception of  TYPEIII message from all the nodes with:
                $nodeID \in InfoExchangeList \wedge nodeID > myID$
        {
            Upon the reception of a TYPE III message {
                IF $myCID > CID$ {
                 $myCID=CID$
                        send to all nodes with $nodeID \in InfoExchangeList$
                        | myID | myCID |
                }
              }
        }
    }
```

4.2 DDC Example

In this section we complete the description of the DDC algorithm by providing an example to demonstrate the functionality of the algorithm and its various phases for the construction of a clustering map. Assume that we have the following network, consisting of 7 nodes:

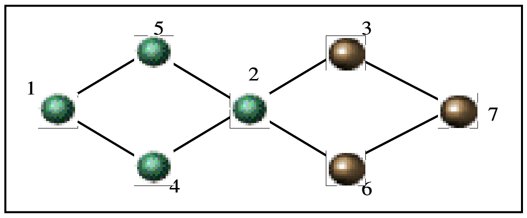


Figure 5. DDC example: Sample Network

If the metric of interest is the relative velocity of nodes, then in Phase I the nodes broadcast their node IDs, their direction and their speed to their 1-hop neighbors. Also, in phase I the nodes collect the same information (node ID, direction and speed) from their 1-hop neighbors. After the nodes have collected the appropriate information then they can compute their relative velocity compared to each one of their 1-hop neighbors. The value of relative velocity $U_{r_{ij}}$ of two 1-hop neighbors (*node*$_i$, *node*$_j$) is the same independently if this is computed separately at *node*$_i$ and *node*$_j$. Assume that for the network of the above figure the relative velocities computed for every pair of 1-hop neighbors are:

Relative Velocity

$U_{r_{ij}} = U_{r_{ji}}$

node ID	1	2	3	4	5	6
2	-					
3	-	2				
4	0.2	0.3	-			
5	0	0.1	-	-		
6	-	2.5	-	-	-	
7	-	-	0.1	-	-	0

node ID

Figure 6. Relative Velocity values as computed from the connected nodes

Based on the above computed values of relative velocity, each node selects the best match of its 1-hop neighbors (e.g. lowest relative velocity). Following this rule the selection of each of the participating nodes by the end of Phase I is:

$$1 \Rightarrow (5) \quad 2 \Rightarrow (5) \quad 3 \Rightarrow (7) \quad 4 \Rightarrow (1)$$
$$5 \Rightarrow (1) \quad 6 \Rightarrow (7) \quad 7 \Rightarrow (6)$$

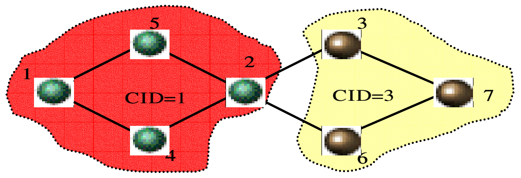
By entering Phase II, the nodes inform their selected 1-hop neighbors. Each node collects and records the IDs of the nodes from which they have been selected. The recorded information from each of the nodes is:

$$1 \Leftarrow (4,5) \quad 2 \Leftarrow (\ ) \quad 3 \Leftarrow (\ ) \quad 4 \Leftarrow (\ )$$
$$5 \Leftarrow (1,2) \quad 6 \Leftarrow (7) \quad 7 \Leftarrow (3,6)$$

By combining the above information with their selections from Phase I each one of the nodes generates an *InfoExchange* list. The *InfoExchange* lists of the nodes by the end of Phase II are:

$$\textit{InfoExchange Lists}$$
$$1:(1,4,5) \quad 2:(2,5) \quad 3:(3,7) \quad 4:(1,4)$$
$$5:(1,2,5) \quad 6:(6,7) \quad 7:(3,6,7)$$

Phase III completes the generation of the clustering map. Each node will utilize the *InfoExchange* list in order to select the cluster to join. The nodes are listening to the clustering selections of the lower ID 1-hop neighbors that belong to their *InfoExchange* list until their turn comes to decide on the cluster to join. After they decide, they wait for the rest of the 1-hop neighbors (e.g., nodes with higher IDs) in their *InfoExchange* list to decide. For the specific network of figure 5, the generated clustering map after the completion of Phase III looks like:



$$CID = 1 : \{1,2,4,5\}$$
$$CID = 2 : \{3,6,7\}$$

Figure 7. The clustering map established from the DDC algorithm

If we carefully observe the values of the relative velocities of the nodes assigned in the same cluster, are much lower compared to the values of the relative velocities of the nodes assigned in different clusters. The DDC algorithm behaves in accordance to our clustering intensions by grouping together nodes with similar mobility characteristics, so we expect that the generated clusters are robust to the topology changes. As we are going to show later in the performance evaluation section, the generated clusters are more robust to topology changes compared to other distributed clustering algorithms that do not take into consideration the dynamics of the network.

5 Performance Evaluation

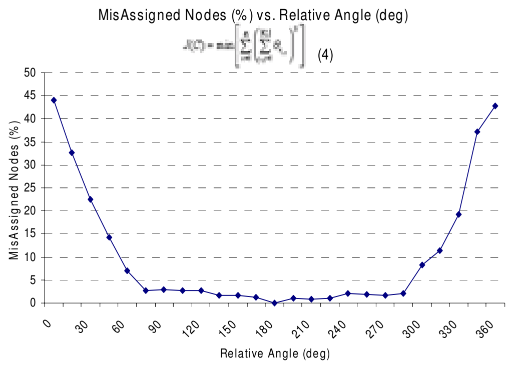
Having described the two proposed clustering algorithms, in this section we will present their performance characteristics. Through the performance evaluation of these algorithms, we want to exploit the optimal quality of clusters generated from the SA algorithm and the effectiveness of DDC algorithm to generate robust clusters in a distributed fashion, based only on 1-hop information. In the former case the optimality of generated clusters is related to the ability of the proposed cost functions to identify successfully the various mobility groups and cluster the corresponding nodes together, in the latter case the utilization of the well defined local mobility metrics for the construction of clusters can still be proven effective in generating clusters that can handle better the topology changes compared to other existing distributed clustering algorithms. The point that we want to highlight through the performance evaluation of both algorithms is that the mobility metrics we propose are successful in meeting our objectives of constructing robust to the topology changes clusters and reducing the imposed clustering maintenance overhead.

5.1 Simulated Annealing and Mobility Cost Functions

The performance evaluation of SA consists of two parts. The first part has to do with the ability of SA along with the application of the proposed cost functions to identify accurately the nodes with similar mobility characteristics and cluster them together. About the second part and since SA is a centralized approach we want to exploit its speed of convergence and specify under what network conditions SA can be applied in practice and still be able to capture the network dynamics. The most important part of the performance evaluation is the first part, because the proof that the proposed cost functions meet our objectives, can help us to identify the most appropriate metrics for the generation of robust clusters.

The effectiveness of the combination of SA with the proposed cost functions is evaluated by setting up carefully the experimental environment. We generated connected networks where the nodes had been assigned in two different mobility groups. Our goal was to test the accuracy of SA and the various cost functions in identifying these two predefined mobility groups. Since we are interested in group mobility we had to select the appropriate group mobility model. In our experiments we utilized the Reference Point Group Mobility (RPGM) Model. In RPGM we define a number of Reference Points (RPs) equal to the number of mobility groups we want to establish. To complete the definition of mobility groups, each node is assigned to a RP. The movement of the nodes is characterized from the mobility patterns of their corresponding RPs. These mobility patterns are assigned manually to the various RPs in the form of trajectories. When a RP moves to a new location each corresponding node is assigned to a random radius and direction around the new position of the RP. Because of the functionality of RPGM model and the randomness in the selection of the new node position, it is obvious that nodes that belong into the same group may have different speeds and directions, which makes our work of identifying the various mobility groups more difficult but makes the evaluation of our clustering approach more general.

The following results were collected by assuming two mobility groups, where the corresponding RPs where moving on a straight line with constant relative direction θ_{RP_1,RP_2} and constant relative speed $S_{RP1,RP2}$. We varied the relative direction from 0^o to 360^o (e.g., $\theta_{RP_1,RP_2} \in \left[0^o \ldots 360^o\right]$) with a step of 15^o. In each run we measured the percentage (%) of nodes that they were assigned in an incorrect cluster. The variation of this percentage versus θ_{RP_1,RP_2} and $S_{RP1,RP2}$ for the various proposed cost functions is given in the following graph:



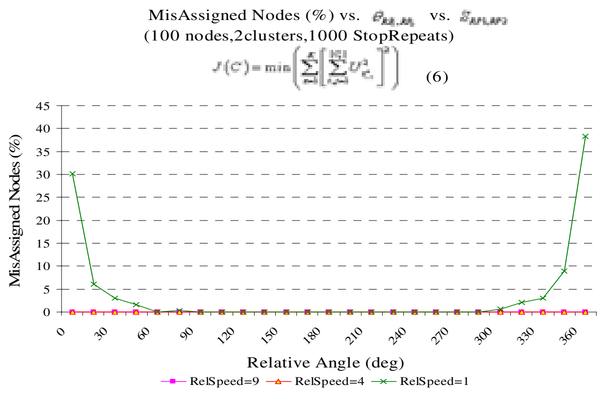


Figure 8. Accuracy of the Combination of SA with the cost function (4) (left) and the cost function (6)(right) for various relative directions and relative speeds of the RPs.

Figure 8 (left), indicates that the cost function (4) can identify accurately the various mobility groups especially when the groups are moving in relative directions such that $\theta_{RP_1,RP_2} \in \left[30^o ... 330^o\right]$. When $\theta_{RP_1,RP_2} \notin \left[30^o ... 330^o\right]$ then the proposed cost function has difficulty to identify the mobility groups. This is not a limitation of the accuracy of the cost function since in this scenario the accurate selection of mobility groups is not restricted to the original mobility groups, because of the similarity in their directions. But still we can do better if we incorporate the speed of the participating nodes into the cost function. By doing so, figure 8 (right) indicates that cost function (6) presents much better accuracy than cost function (4) which depends solely on the nodes direction. For $S_{RP1,RP2} > 1\ m/s$ the mobility groups are identified with accuracy 100%. The latter result proves the effectiveness of the cost function (6) and the optimality of the decisions taken from SA algorithm.

Even though the effectiveness of the proposed mobility cost functions can be exploited from the above results, due to the centralized nature of SA we have to evaluate the speed of convergence of the method and decide if and when it is applicable in dynamic environments. The following graph presents the convergence time of the algorithm for networks of various sizes. The following convergence times of SA were collected by running the algorithm on a LINUX box (CPU: 700MHz, RAM: 256MB) with the help of the *gettimeofday(*struct timeval **tv*, struct timezone **tz)* function that exists under the UNIX/LINUX systems.

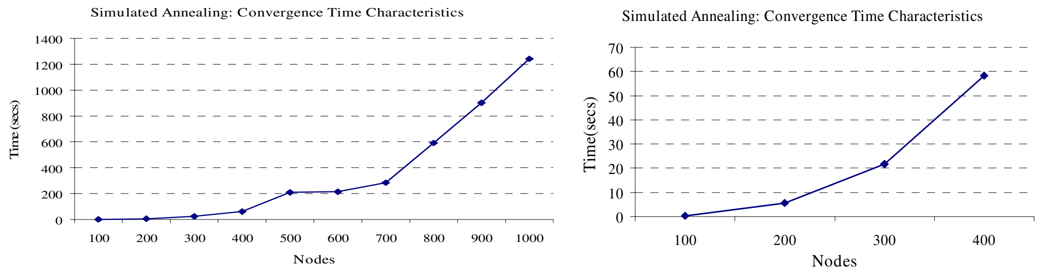


Figure 9. SA Convergence Time Measurements for various network sizes

It can be observed that the completion time of the general SA optimization increases significantly as the network size increases. The completion time is 5s for 200 nodes and in the order of 20s for network sizes of 300 nodes; this time is increased to 1200s for network sizes of 1000 nodes. Thus, we can conclude that SA can be applied either a) in small networks (e.g. few hundreds of nodes), or b) large networks with low rate of topology changes (e.g., sensor networks where we have none or very slow movement of the nodes). In order to SA to be useful in large dynamic environments, convergence time enhancements are required so that the SA optimization can produce faster, clustering maps of high quality.

Even though the above graph reveals the speed of the algorithm, we cannot use it alone to identify the network scenarios where the algorithm could potentially applied. The following graph presents the acceptance ratio of the clustering decisions taken by SA by the time the algorithm terminates. This is an important graph because as SA performs the optimization the network topology changes, so we need to know if the generated clustering decisions are still valid for the topology that the network presents by the time the SA converges to these decisions.

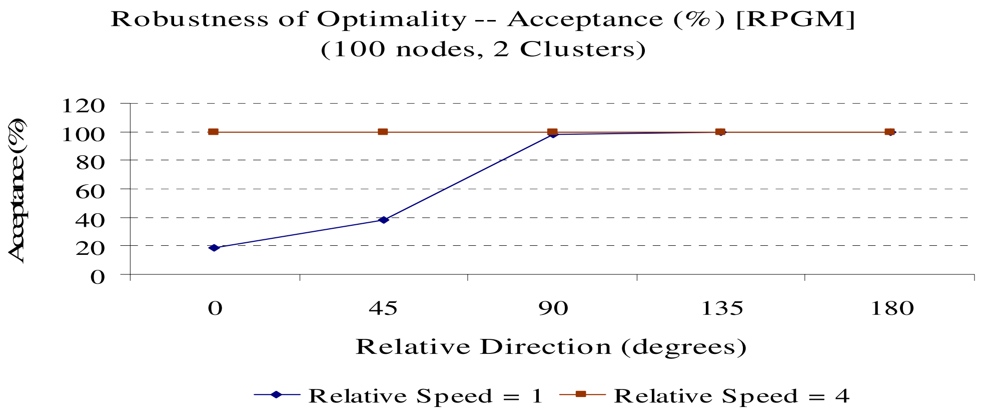


Fig.10. Acceptance (%) of the clustering decisions by the termination of SA subject to RPGM model

Figure 10 above, shows that whenever the algorithm identifies the mobility groups with high accuracy, then due to the fact that the members of these groups remain connected for long periods of time then the clustering decision are more likely to be feasible by the time SA terminates. In cases where distinct mobility groups present similar mobility characteristics and are topologically close, SA cannot identify the mobility groups with high accuracy and this results in clustering decisions that may not be feasible by the time SA terminates. The main observation is that SA can be applied in cases where there are clearly defined mobility groups (e.g., present distinct mobility characteristics) even though SA is slow in nature. The latter observation is one more proof that the robustness of clustering decisions makes them feasible for long periods of time.

5.2 Distributed Algorithm

The results we collected for the SA algorithm prove the effectiveness of the mobility metrics we consider for the generation of robust clusters. Since SA cannot capture the dynamics of a large mobile network, we introduce the distributed algorithm, which has been designed to assign nodes with similar mobility characteristics into the same cluster. Since DDC is a heuristic algorithm we do not expect that the clustering decisions will have the quality of the ones taken by SA, but our goal in this case is to trade off the quality of clustering with the speed of convergence. On the other hand, by applying the same mobility metrics, we expect that the clusters generated by DDC, present better robustness characteristics compared to other existing distributed clustering algorithms.

We compared the robustness of DDC with the lower ID algorithm [2]. The latter algorithm selects cluster heads based on the node IDs and then assign nodes to these clusterheads based on the hop distance (e.g., nodes are assigned to the lower ID clusterhead which is 1–hop away). Obviously, the lower ID algorithm does not take into consideration the dynamics of the network and also the generated clusters have at most 2-hops diameter.

Since DDC was designed for the generation of clusters that are robust to the topology changes, the performance metrics of interest we selected for its evaluation and comparison with the lower ID algorithm, are the average membership changes and the average number of generated clusters. We applied both DDC and lower ID algorithms in network environments of various sizes and mobility levels. Namely, we generated networks of 100 to 1000 nodes and we applied the Random Walk model (e.g., Random Waypoint model with pause time equal to 0). In Random Walk model the nodes select at random destinations which are always in the limits of a pre-specified area and they are moving to these destinations with constant speed, which is also selected at random between 0 and a pre-specified maximum value. When the nodes reach their destinations, they immediately select new destinations and new speed and the process is repeated. We varied the maximum allowable node speed between 1m/s and 10m/s to test the ability of DDC to generate robust clusters even in scenarios of high mobility. We ran the algorithms for 1000s of network time and we were evaluating the performance metrics of interest per 1s. The following graph represents the average membership changes for networks of size 100 to 1000 nodes and maximum allowable node speeds of 1m/s to 10m/s.

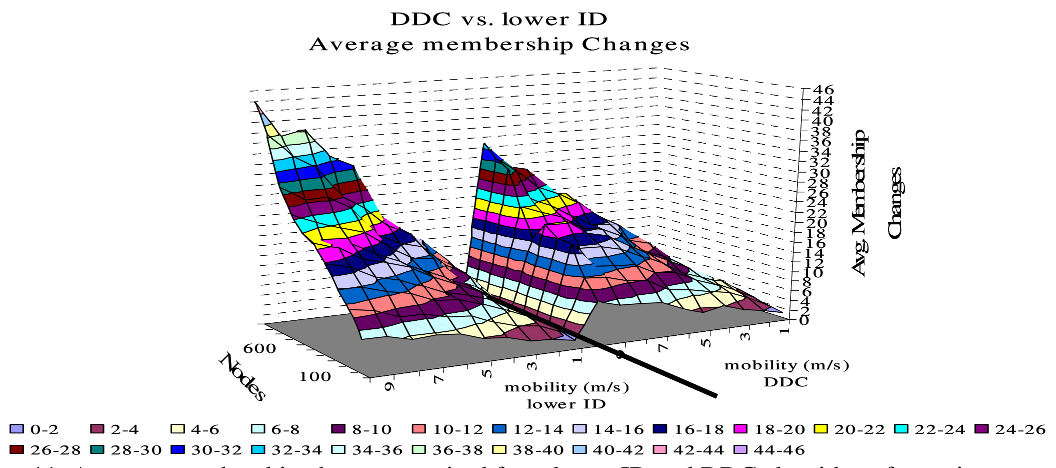


Figure 11. Average membership changes required from lower ID and DDC algorithms for various network sizes and mobility levels

The left part of the graph represents the average membership changes of the lower ID algorithm and the right part represents the average membership changes of the DDC algorithm, respectively. The higher the mobility and the larger the size of the network, the better the performance of DDC algorithm compared to the performance of the lower ID algorithm, subject to the average membership changes. For example for 1000 nodes and 10m/s maximum speed, DDC requires 32 membership changes per second and lower ID requires 44 membership changes per second. For 1000 seconds of network time, DDC requires on average 12000 less membership changes than lower ID algorithm, which is an improvement of 27.2%.

The characterization of the robustness of the clustering maps can be more complete if we compare the number of clusters that are generated from DDC and lower ID algorithms. The following figure represents the average number of clusters generated from each one of the algorithms.

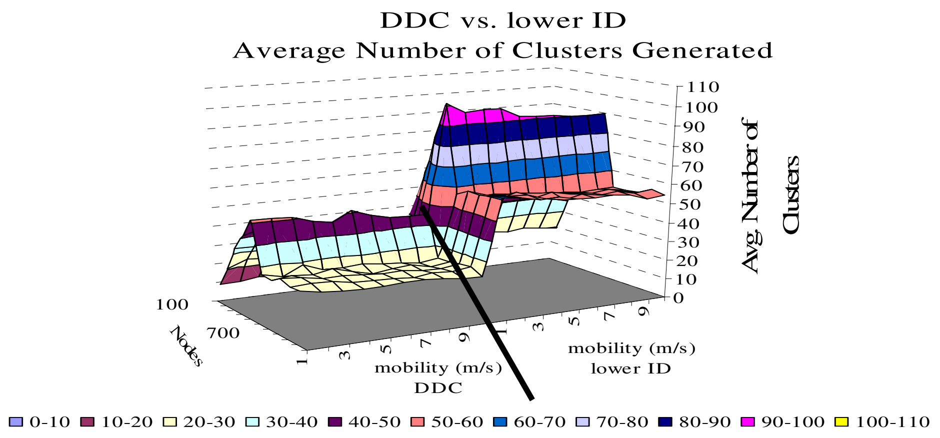


Figure 12. Average number of clusters generated from lower ID and DDC algorithms for various network sizes and mobility levels

The left part of the graph represents the average number of clusters generated from the DDC algorithm and the right part represents the average number of clusters generated from the lower ID algorithm, respectively. The general observation is that the number of clusters that the lower ID generates is more than double the number of clusters generated from DDC. Following the latter observation, the cluster size of DDC is more than double the cluster size of lower ID, which was expected since lower ID generates clusters of 2-hops diameter at most. For large mobile networks, the behavior of lower ID it is more possible to harm the performance of the network instead of improving it, because of the large number of clusters and the large number of membership changes. The approach of DDC to generate larger clusters makes the clustering map more vulnerable to the mobility, but still DDC due to its clustering philosophy presents

more stable clustering maps than lower ID algorithm. The hierarchy generated from DDC is more likely to improve the performance of the network due to the significant reduction of maintenance overhead.
The stability of the clustering map generated from DDC can be even better, since we evaluated the algorithm subject to the Random Walk model, which is not a group mobility model. If we evaluate DDC algorithm subject to Reference Point Group Mobility model or another group mobility model we will get much better results in terms of the robustness of the generated clustering maps. DDC was designed to identify and cluster together nodes with similar mobility characteristics. So, in a network environment where distinct mobility groups exist, DDC tends to accurately identify these groups by clustering the corresponding nodes together.

6 Conclusions

In this paper we presented two approaches for hierarchy generation in dynamic environments, the centralized approach which is based on SA and the distributed one which tries to resemble the quality of the clustering maps generated by the centralized approach. Both approaches are based on a set of mobility metrics, which are applied for the generation of hierarchy. Our objective was to generate clustering maps that are robust to the topology changes. The performance evaluation of the centralized approach proves the effectiveness of the proposed mobility metrics in identifying accurately the various mobility groups. Even though SA is slow for large and highly dynamic networks, its clustering philosophy establishes the algorithm applicable in scenarios where there are distinct mobility groups, with stable membership. Also, the algorithm is applicable in more dynamic cases than the latter, but for small to medium size networks (up to 300 nodes) which present moderate mobility characteristics.

The functionality of the Dynamic Distributed Algorithm (DDC) is based on the same mobility metrics as the centralized algorithm. The difference is that the available information is local (1-hop information) and based on this information we require the generation of robust clusters. We evaluated the algorithm subject to the Random Walk mobility model for various network sizes and mobility levels and we compared the average membership changes and the average number of generated clusters of DDC with the corresponding performance metrics of the lower ID algorithm. The collected results proved the robustness of the clustering maps generated from DDC, even though were based on a mobility model that does not favor group mobility. Namely, DDC requires 27.2% less membership changes than the lower ID algorithm even though it generates two times larger clusters.

Currently we are investigating ways to improve the convergence time of SA algorithm, so it can be applicable in more dynamic networks. Also, we are moving towards the direction of generating hybrid clustering algorithms that combine SA and DDC, due to the fact that both algorithms operate based on the same metrics and have the same clustering objectives, so we can combine the optimality of SA with the speed of DDC. The proposed algorithms are generic and can be used independently of the specified metrics, for that reason we investigate metrics that are not restricted only to the mobility of the nodes but also to their power management.

References

[1] K. Manousakis, J. McAuley, R. Morera, J. Baras, "Routing Domain Autoconfiguration for More Efficient and Rapidly Deployable Mobile Networks," Army Science Conference 2002, Orlando, FL

[2] R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," IEEE Journal on Selected Areas in Communications, pages 1265-1275, September 1997

[3] D. Baker, A. Ephremides, and J. Flynn "The design and simulation of a mobile radio network with distributed control," IEEE Journal on Selected Areas in Communications, SAC-2(1):226--237, 1984

[4] M. Chatterjee, S. K. Das, D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks, " Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks), Vol. 5, No. 2, April 2002, pp. 193-204

[5] S. Basagni, "Distributed and Mobility-Adaptive Clustering for Multimedia Support in Multi-Hop Wireless Networks," Proceedings of Vehicular Technology Conference, VTC 1999-Fall, Vol. 2, pp. 889-893

[6] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, Optimization by Simulated Annealing, Science 220 (13 May 1983), 671-680

[7] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization," in E. H. Aarts and J. K. Lenstra (eds.), "Local Search in Combinatorial Optimization," John Wiley and Sons, Ltd., pp. 215-310, 1997