Applying Simulated Annealing for Domain Generation in Ad Hoc Networks

Kyriakos Manousakis
Electrical and Computer Engineering and
the Institute for Systems Research
University of Maryland, College Park
College Park, MD, 20740
kerk@isr.umd.edu

Anthony J. McAuley, Raquel Morera
Telcordia Technologies Inc.
Piscataway, NJ, 08854
{mcauley, raquel}@research.telcordia.com

Abstract[i]— **If heterogeneous ad hoc battlefield networks are to scale to hundreds or thousands of nodes, then they must be automatically split into separate network domains. Domains allow routing, QoS and other networking protocols to operate on fewer 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]. Domain generation (or clustering) can be done using either local or global information. The two approaches are complementary since local domain generation reacts faster, requires less overhead, and is more robust; while global domain generation provides better overall domains. While most existing work has concentrated on local distributed solutions, this paper reports on new global domain generation techniques. In particular we concentrate on the design of *good cost functions* and *efficient optimization algorithms*. We show that simple "intuitive" cost functions do not produce good domains; rather we need complex functions with multiple parameters depending on the design goals (e.g., low overhead or low delay). Although existing optimization algorithms are too slow to be useful in a large dynamic network, we show that a modified simulated annealing algorithm, with well chosen cooling schedule, state transition probabilities and stop criteria, produces good quality domains in acceptable time.**

Keywords-ad hoc networks; mobile networks; dynamic clustering; simulated annealing

I. INTRODUCTION

In recent years there has been an increasing interest in ad hoc networks that do not rely on a fixed infrastructure. The ability to deploy these networks quickly and have them work through rapid changes makes them ideal for battlefield and emergency situations. There have been many good solutions proposed to deal with topology management, autoconfiguration, routing and QoS in ad hoc networks; however, most of these solutions do not scale well (e.g., only to about 50 nodes). To build ad hoc networks with hundreds or even thousands of nodes, such as that required for the Future Combat System (FCS), the network must be split into relatively independent layer 3 clusters or domains.

We assume the creation of layer 3 clusters or domains is done after layer 2 topology management has set local parameters such as the link frequencies, spreading code, transmit power and antenna direction. At this point, when the ad hoc network is simply an interconnected mesh of potentially thousands of nodes, the domain generation divides node interfaces into different layer 3 domains. The domain generation then continuously adjusts the domain as nodes and links change to maintain good network performance.

Smaller domains allow routing, QoS and other networking protocols to operate on fewer nodes, with cross-domain interaction only through a few border nodes. This division has two key benefits. First, it reduces overall protocol overhead. In most routing protocols, for example, the route update overhead grows as $O(n^2)$ as the number of routers n in a domain increases. Using smaller domains, with inter-domain interaction through a single border router per domain, we can reduce overall overhead to $O(n\log n)$. Second, if the domains are well designed, then networking protocols can be tuned to more homogenous conditions. For example, if part of the network has links constantly going up and down, then it can be put in a separate routing domain whose border router does not propagate internal changes.

The domain generation and maintenance can be done using either local or global information. The two approaches are complementary since local domain generation reacts faster, requires less overhead, and is more robust; while global domain generation provides better overall domains. Most existing work on domain generation, however, has used only very limited local information. Indeed, most approaches simply elect a "cluster-head" within each subnet based on node attributes like the lowest ID or highest degree [2] [3]. Some proposals use local metrics during cluster generation, but the metrics are utilized just for the selection of cluster-heads; the generation of clusters is based only on the distance in hops of nodes from the corresponding cluster-heads [4] [5].

Our goal is to take global network environment and its dynamics into account, then optimally grouping together nodes

[i] Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance (CTA) Program, Cooperative Agreement DAAD19-2-01-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

based on the appropriate metrics to improve *a priori* selected aspects of the performance of the network. A key question in this work is whether the domains generated using global information significantly improves the network's performance. If not, then the extra overhead will degrade the performance of the network, and we should simply perform robust local domain generation. This paper reports on the two key aspects of global domain generation that we are needed to generate significantly better network performance even with hundreds of dynamic nodes: designing **good cost functions** and **fast optimization algorithms**.

Section 2 investigates the cost functions for defining good domains. Section 3 describes the general Simulated Annealing (SA) algorithm for clustering and Section 4 shows how a modified SA algorithm can produce near optimal clustering maps quickly. Section 5 evaluates the clustering maps generated by the SA algorithm with different cost functions. Finally, section 6 gives our conclusions and some future directions.

II. COST FUNCTIONS

This section looks at the design of cost functions that determine the goodness of a particular domain configuration map.

A. *Metrics*

We treat each cluster as part of a group and not as an individual entity in order to optimize the overall network performance and not just on the performance of a single cluster. The key metric used to decide the best domain configuration is the global topology map. This IP topology simply gives information about who can talk to whom, and through which interfaces. In addition to topology it may be beneficial to use a lot of additional metrics, such as: a) Link performance (e.g., capacity, delay, error rate, and up-time); b) Node velocity (e.g., average node speed and direction); c) Groups (e.g., what unit of action or brigade the node or user is a member of); d) Mission information (e.g., giving some indication of likely mobility or traffic pattern)

Although we believe these additional metrics can potentially be a very powerful tool in creating better topologies, this paper only considers the use of basic topology information. It is important to look at this simple case first, since: a) other metrics might not be available, b) may not be estimated with accuracy in an ad hoc network (and may thus harm the instead of helping the network), and c) can be done before a node is configured with it IP information (e.g., address and routing protocol).

B. *Cost Function Variables*

We designed cost functions for the generation of clusters based on the following variables:

- *Diameter* (***D(C$_i$)***)- The size of the longest path within a cluster in number of hops. Minimizing diameter is useful because it can reduce overhead and reduce the latency of many networking protocols. For example, a proactive routing protocol, which exchanges routing information among all nodes, can update information quicker (e.g., due to link failure) and using less total hops if the diameter is smaller.
- *Number of nodes* (**N(*C$_i$*)**)- The number of nodes that have been assigned to the cluster. The importance of Cluster Size metric rises from the fact that the overhead and performance of most networking protocols is strongly dependent on the number of nodes. For example, we know that the overhead of most routing protocols is proportional to the square of the number of nodes
- *Number of Border Nodes* (***B(C$_i$)***)– The number of nodes that interconnect two or more clusters. There are scenarios where we want to have some minimum number of border nodes to improve robustness or to provide more bandwidth for inter-domain communication. In other cases we want to minimize the number of border nodes, in order to isolate each cluster from the rest of the clusters in the network (i.e. security, small number of inter-cluster connections).

C. *Single Variable Cost Functions*

- Based on Diameter

The simplest cost function E based on the cluster diameter $D(C_i)$ is to sum the diameters of each of the K clusters C_i with our objective being to generate clusters such that the Cluster Diameter is minimized among all clusters.

$$E = \min \sum_{i-1}^{K} D(C_i) \tag{1}$$

Through our testing, we found that this simple formula (1) produces very unbalanced clusters in terms of their diameter.

$$E = \min \sum_{i-1}^{K} \left(D^2(C_i)\right) \tag{2}$$

$$E = \min\left(Var\left(D^2(C_1), D^2(C_2), \ldots, D^2(C_K)\right)\right) \tag{3}$$

These functions (2) and (3) generate balanced diameter clusters.

- Based on Cluster Size

The simplest cost function based on the number of members in each cluster is to sum the number of nodes in each of the K clusters with our objective being to generate clusters such that the cluster sizes are balanced:

$$E = \min \sum_{i-1}^{K} N(C_i) \tag{4}$$

Similarly with (1) this function produces unbalanced size clusters, so we applied more complex functions (e.g., (5), (6) and (7)), which as we present in section 5 result in balanced size clusters. Cost function (5) sums the squares of the size of each cluster.

$$E = \min \sum_{i-1}^{K} N(C_i)^2 \tag{5}$$

The goal of producing balanced size clusters is useful if we select K so as to produce the "right sized" clusters, assuming we can balance them using this cost function. An alternative we investigated was to use variance (6) or to put in explicit information about the optimal number of nodes (*OptSize*) of a cluster (7).

$$E = \min\left(Var\left(N^2(C_1), N^2(C_2), \ldots, N^2(C_K)\right)\right) \tag{6}$$

$$E = \min\left(\sum_{i-1}^{K} N(C_i)^2 + \sum_{i-1}^{K}\left(N(C_i) - OptSize\right)^4\right) \tag{7}$$

- Based on Border Nodes

The final simple cost function we tested was to use the number of border nodes. Cost function (8) shows example of a cost function that attempts to minimize the number of border nodes *B(C)*.

$$E = \min\sum_{i-1}^{K} B(C_i) \tag{8}$$

D. *Multiple Variable Cost Functions*

In addition to the simple cost functions, we also investigated cost functions that combine multiple variables. Equations (9) and (10) are some examples, where we combine the size of the clusters with the number of border nodes and the diameter of the clusters with the number of border nodes respectively.

$$E = \min\left(Var\left(N^2(C_1), N^2(C_2), \ldots, N^2(C_K)\right) + \left(\sum_{i-1}^{K} B(C_i)\right)\right) \tag{9}$$

$$E = \min\left(Var\left(D^2(C_1), D^2(C_2), \ldots, D^2(C_K)\right) + \left(\sum_{i=1}^{K} B(C_i)\right)\right) \tag{10}$$

III. Simulated Annealing

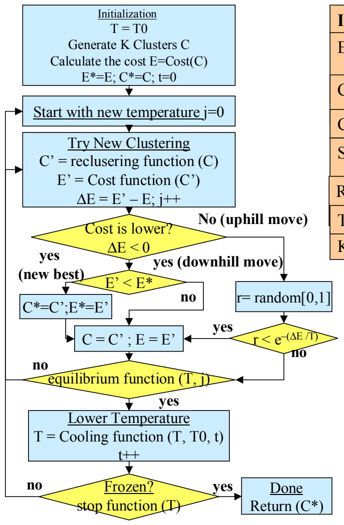
We here describe the simulated annealing algorithm applied to the specific networking problem of partitioning a network in *K* disjoint clusters.

A. *General SA Algorithm*

Simulated annealing (SA) has been widely used for tackling different combinatorial optimization problems [6]. 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, T0, 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*. In order to avoid local minima, the algorithm randomly accepts a new configuration C' even though E' is greater than E*. 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. The lower the temperature, the lower the probability of accepting worse configurations.



Inputs	Examples
Equilibrium unction	Constant (j = 5000); "Stop repeats"
Cost function	$\sum^{K} Diameter(C_i)$
Cooling function	Geometric or logarithmic
Stop function	Minimum temperature "Stop repeat" criteria
Reclustering	Random move of one node
T0	Initial Temperature
K	Number of clusters

Variable	Definition
T	Current Temperature
C	Current Cluser map
C'	New cluser map to test
C*	Champion cluster map
E	Current cost
E'	Cost of new cluster map
E*	Champion cost
j	Inner loop counter
t	Outer loop counter

Figure 1. Simulated Annealing Algorithm for network partitioning

B. *Comparison of SA with Other Optimizations*

SA is one of many choices when looking for optimization algorithms. For example, one can use the 2-Opt, 3-Opt, Lin-Kernighan, genetic and neural net algorithms. As shown in the work of Johnson [7] depending on the problem to which it is applied, SA appears competitive with many of the best heuristics: a) easy to implement; b) provides good solutions for most problems; c) can deal with cost functions with arbitrary degrees of non-linearity, discontinuity, and randomness; d) can process arbitrary boundary conditions and constraints imposed on these cost functions.

However, the general SA optimization can be quite slow. Figures 2a and 2b show the completion time of SA for various network sizes (25 nodes to 1000 nodes) and fixed cluster size (25 nodes). The number of clusters to be generated varies based on the network size and Cluster size (7) has been taken as the cost function.

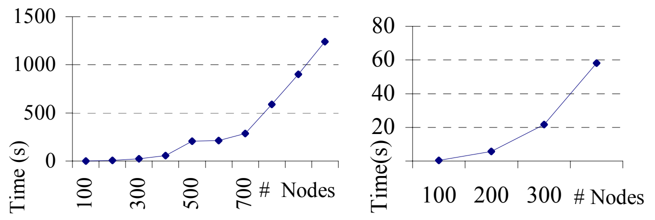


Figure 2. Convergence Time vs. Number of Nodes. (Cost Function= (6), Uniform Transition Probs)

It can be observed that the completion time of the general SA optimization increases significantly as the network size increases. The completion time is in the order of 50s for network sizes of 300 nodes; this time is increased to 400s for network sizes of 1000 nodes. Thus, we can conclude it need either a) a 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, we propose enhancement to the general SA optimization to produce good clustering maps in a much shorter period of time.

IV. Improving the Simulated Annealing Algorithm for Clustering

The following sections describe some of the enhancements that we propose to decrease SA's computational time while keeping its optimality. We focus on three functions from figure 1: a) the function that evaluates the frozen state (**stop function**), b) the cooling scheme (**cooling function**) and; c) algorithms that generate new clusters at each iteration of SA (**reclustering function**).

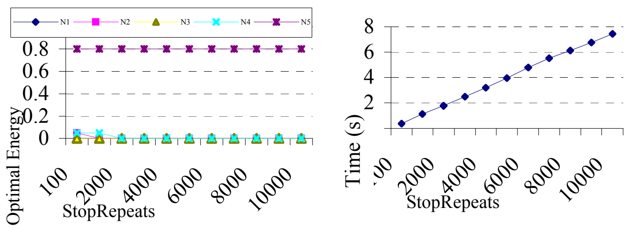


Figure 3. (left) Opt. Energy vs. StopRepeats , (right) Time vs. StopRepeats (Nodes=100,Clusters=5) (Cost Function=(6))

A. *Improving the Stop Function*

In our implementation, SA terminates when the same optimum cost value E^* is observed for a continuous number of iterations, called *StopRepeats.* Figure 3 (left) shows E^* as a function of the *StopRepeats* parameter for 5 different 100 node networks. We can observe that in all scenarios we get similar E^* values for values of *StopRepeats* larger than 100. Therefore, the optimality of the algorithm is the same whether the algorithm is stopped using a number of *StopRepeats* equal to 100 or equal to 10000. The smaller the number of *StopRepeats*, the faster the completion time of the algorithm (fig. 3 right).

B. *Improving the Cooling Functions*

The speed of the SA algorithm depends on the cooling schedule used to lower down the temperature at each iteration. The two most popular cooling schedules are:

- Logarithmic Cooling Schedule : $T_t=T_0/(1+\ln t)$
- Geometric Cooling Schedule : $T_t=\alpha^t T_0$

T_t, T_0 are the new and initial temperatures, respectively and t is the number of iterations.

Figure 4 shows the effect the cooling schedule has on the progress (value of cost function) and the speed (number of iterations) of SA on a network of 100 nodes.

It can be observed that a geometric cooling schedule decides faster on the clustering map than a logarithmic cooling schedule. Therefore, it is better suited to a dynamic environment. However, the clustering map obtained with the geometric cooling scheme may not be as optimal as the one obtained with the logarithmic cooling scheme. Nevertheless, as figure 6 (right) shows this is not necessarily true in every case.

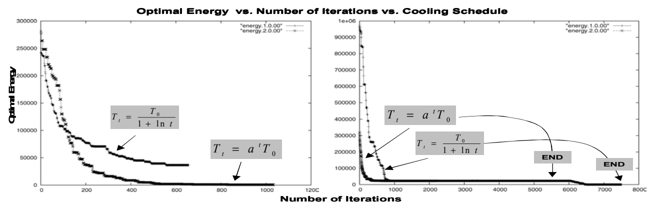


Figure 4. Optimal Energy vs. Number of Iterations vs. Cooling Schedule

C. *Improving the Reclustering Function*

We can further improve the convergence time of the SA if we bind the state transition probabilities, used for the generation of new clustering maps, to the cost function. Given a cluster map C, a new clustering map C' is generated by selecting a node from a cluster ("from" cluster) and moving it to a different cluster ("to" cluster). In figure 5 we show the convergence time of SA for two different clustering map generation schemes:

1) Initially, for simplicity and generality, we use uniform probabilities to select nodes in the "from" and "to" cluster.

2) We customized the transition probabilities to the cost function. For example, for a cost function that aims for balanced size clusters, we assigned higher probabilities to those clusters with larger cluster size in order to select the "from" cluster and higher probabilities to those clusters with smaller size to select the "to" clusters. This approach utilizes the following state transition probabilities:

- $$P(\text{"from" cluster } i) = \frac{|C_i|}{(\text{Total Number Of Nodes})} \qquad (11)$$
- $$P(\text{"to" cluster } i) = \frac{(\text{Total Number Of Nodes}) - |C_i|}{(\text{Number of Clusters} - 1)\text{x}(\text{Total Number Of Nodes})} \qquad (12)$$

Figure 5 compares the completion time of the SA algorithm applying uniform and non-uniform state transition probabilities described by (11) and (12) for a network of 200 nodes.

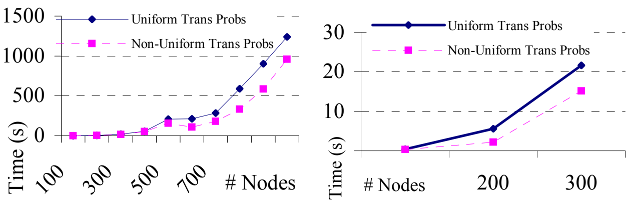


Figure 5. Convergence Time: Uniform vs. Non-uniform probabilities Uniform vs. Non-Uniform Trans Probs (Cost Function=(6))

Thus, where the search space is large (e.g., small number of generated clusters imposes larger number of possible solutions using non-uniform probabilities improves the speed of the algorithm up to 100%. With smaller search space, however, the application of non-uniform state transition probabilities does not improve the speed of the algorithm compared to the uniform probabilities. As our main objective is the improvement of the speed of the SA algorithm when we have large search space, state transition probabilities customization approach is desirable.

D. Summary of SA Options for Clustering

Since the dynamics change the topology, we do not want the optimal clustering map but a good quality clustering map as fast as possible. Thus, we chose a **geometric cooling** schedule since it gives good quality clustering maps fast (even though in some cases may not be as good as the results we get from the logarithmic cooling schedule). We also set the ***StopRepeats* to the lowest value without significantly affecting the protocol optimality** and **take into account the cost functions in the generation of new clusters**.

V. Evaluation of Cost Functions in Dynamic Environments

This section evaluates the clustering maps that the SA algorithm generates with the application of the cost functions described in section 2. In order to test the proposed cost functions and evaluate the performance of SA algorithm we require a number of networks (connected graphs). For the construction of the inputs we implemented software that produces random connected graphs given the number of nodes, their transmission range and the area. With this set of input parameters we can control the density of the network. Figure 6 shows some samples of the networks. The top plot shows the clustering map based on the balanced cluster size (equation (6)) for a network of 100 nodes and 10 clusters.

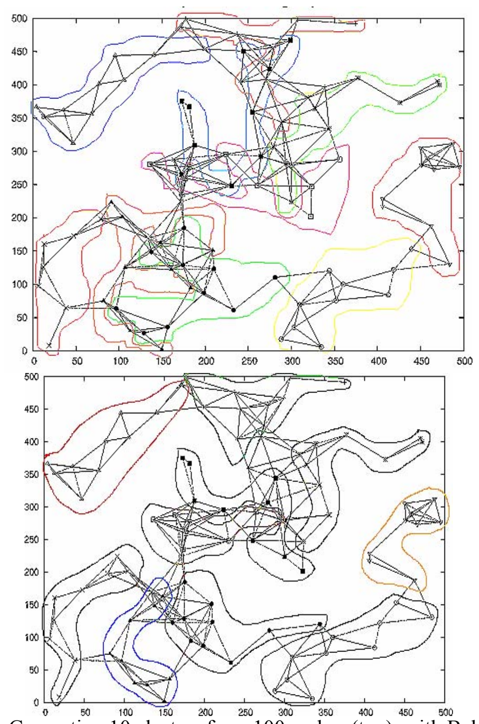


Figure 6. Generating 10 clusters from 100 nodes: (top) with Balanced size clusters, (bottom) minizing the number of Border Routers.

Each closed curve represents a different cluster. By measuring the number of nodes that each cluster has we observe that we have perfectly balanced clusters (e.g., each cluster consist of exactly 10 nodes (10nodes x 10clusters = 100nodes)). However, the clusters overlap.

If we do not want such overlap among clusters, we utilize the metric that minimizes and balances the number of border routers. The bottom of Figure 6b shows the clustering map for equation (9). The results show that we kept the balanced clusters (e.g., 10 nodes per cluster for 10 generated clusters), but with more isolated and topologically defined clusters.

VI. Conclusions

This paper proposes some new cost functions and an optimization algorithm for network partitioning based on Simulated Annealing. We show that simple “intuitive” cost functions, such as minimizing the average domain diameter, do not produce good domains; rather we need complex functions (e.g., equation 9) produce good results. We also show that algorithms that find the optimal configuration are too slow to be useful in a large network with rapidly changing topology, but a modified simulated annealing (SA) algorithm, produces good quality domains quickly. Specifically we use SA with geometric cooling schedule, low StopRepeat value and generate new clusters based on the cost functions.

We have to evaluate the overhead that the centralized approach imposes on the network. Since we deal with mobile networks, the generation of an effective clustering map it is a solution for a limited time window (e.g., depending on the network dynamics) and it cannot provide us with a complete solution for the lifetime of the network. To achieve the maintenance part of the clustering and due to the centralized nature of SA we are currently looking ways to bind the SA with localized algorithms (e.g., distributed heuristics). Finally we think it is important to look beyond topology for input into the global optimization. For example, investigate the use of link quality or a node’s mobility characteristics[ii].

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

[ii] The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government