Optimizing Clustering Algorithm in Mobile Ad hoc Networks Using Simulated Annealing

Damla Turgut
School of EECS
University of Central Florida
Orlando, FL 32816-2450
turgut@cs.ucf.edu

Begumhan Turgut, Ramez Elmasri and Than V. Le
Dept of Computer Science & Engineering
University of Texas at Arlington
Arlington, TX 76019-0015
{bturgut,elmasri,thle}@cse.uta.edu

Abstract— **In this paper, we demonstrate how simulated annealing algorithm can be applied to clustering algorithms used in ad hoc networks; specifically our recently proposed *weighted clustering algorithm*(WCA) is optimized by simulated annealing. As the simulated annealing stands to be a powerful stochastic search method, its usage for combinatorial optimization problems was found to be applicable in our problem domain. The problem formulation along with the parameters is mapped to be an individual solution as an input to the simulated annealing algorithm. Input consists of a random set of clusterhead set along with its members and the set of all possible dominant sets chosen from a given network of N nodes as obtained from the original WCA. Simulated annealing uses this information to find the best solution defined by computing the objective function and obtaining the best fitness value. The proposed technique is such that each clusterhead handles the maximum possible number of mobile nodes in its cluster in order to facilitate the optimal operation of the MAC protocol. Consequently, it results in the minimum number of clusters and hence clusterheads. Simulation results exhibit improved performance of the *optimized* WCA than the original WCA.**

I. Introduction

Mobile ad hoc networks have gained tremendous importance over the last decade. Even though, ad hoc networks were originally designed and developed for military purposes, they are also currently used in various civilian applications such as conferencing, emergency services, home networking, automotivePC interaction, personal area networks and bluetooth, embedded computing applications, and so on [10]. These types of networks are created dynamically in an *ad hoc* manner as the name refers. Their life duration is generally short; it can vary from hours to days to weeks. They become visible in places where a wired backbone is not feasible due to time or economics constraints. An ad hoc network is not only created dynamically but also adapts itself to ever changing network configurations. Analogous to *cell* structure in cellular architecture, the nodes in the network are generally partitioned to individual clusters. Each cluster is managed by a special node called a *clusterhead*. The clusterheads are responsible for the formation of clusters, maintenance of the topology of the network and resource allocation to all of the nodes in their clusters. The set of clusterheads is known as a *dominant set*. Since the configuration of the clusterheads are constantly changing due to dynamic nature of the mobile nodes, creating a large overhead, minimizing the number of clusterheads becomes essential.

Optimization of the clusterheads is an NP-hard problem [1], [2]. A general approximation algorithm that runs in polynomial-time needs to be used in order to solve this kind of problems. It is difficult to find such algorithm to obtain near optimal solution. Simulated annealing is considered an approximation algorithm where it is applicable to various problems in general. Genetic algorithm is another approach in optimizing the cluster election procedure [12]. We choose *simulated annealing* (SA) technique which is a probabilistic search method widely used in wide range of areas such as mathematics, computer science, engineering, operations research, chemistry, physics, and so on [6], [8], [11]. The SA algorithm can be considered as a version of an "iterative improvement algorithm" which considers only specific transitions and terminates in the first local minima found. Unlike this algorithm, simulated annealing allows various types of transitions in which some of them may be opposite towards achieving the goal. For instance, cost-increasing transitions are also accepted along with cost-decreasing transitions whereas iterative improvement algorithm would allows only cost-decreasing ones to pass. However, it is proven that eventually simulated annealing produces more optimal solution than the original iterative improvement algorithm.

In this paper we are focusing on the application of the SA approach to combinatorial optimization problems.

0-7803-7700-1/03/$17.00 (C) 2003 IEEE

Metropolis algorithm [9] was the original idea behind the optimization technique of SA. Kirkpatrick et. al, [7] has used Metropolis algorithm as a global optimizer. Thus, simulated annealing is also known as *global optimizer*. This algorithm is then applied to the physical design of computers. The advantage of using simulated annealing is its ability to scale for large scale optimization problems and its robustness towards achieving local optima convergence.

SA starts with an initial solution, s. A neighbor to this solution s', is then generated as the next solution by SA and the change in cost, $\Delta F(s, s')$ is evaluated. If a reduction in cost is found, the current solution is replaced by the generated neighbor, otherwise we decide with a certain probability whether s remains or s' becomes the current solution. The probability of accepting a transition that causes an increase, ΔF, in the cost is usually called the **acceptance function** and is set to $e^{(\frac{-\Delta F}{T})}$ where T is the control parameter that corresponds to temperature in the analogy with the physical annealing process. In SA, the algorithm is started with a relatively high value of T, to have a better chance to avoid being prematurely trapped in a local minimum. The control parameter is lowered in steps until it approaches to zero. After termination, the final configuration is taken as the solution of the problem at hand. That is, simulated annealing is a generalization of the local search algorithm.

The rest of the paper is organized as follows. In section 2, we propose the optimized version of WCA. Simulation results are presented in section 3 and conclusions are offered in section 4.

II. Optimizing a Clustering Algorithm

Let us briefly summarize the Weighted Clustering Algorithm (WCA). In WCA, the clusterheads are selected based on the weight W_v of node v. W_v is obtained as

$$W_v = w_1\Delta_v + w_2 D_v + w_3 M_v + w_4 P_v.$$

where Δ_v is the *degree-difference*, D_v is sum of the distances of the members of the clusterhead, M_v is the average speed of the nodes, and P_v is the accumulative time of a node being a clusterhead. w_1, w_2, w_3 and w_4 are the corresponding *weighing factors* such that $\sum_{i=1}^{4} w_i = 1$. The node v with the minimum W_v is chosen to be the clusterhead. Once a node becomes the clusterhead, neither that node nor its members can participate in the cluster election algorithm. The election algorithm will terminate once all the nodes either become a clusterhead or a member of a clusterhead. All the clusterheads are aware of their one-hop neighbors as well as the ordinary nodes know their clusterheads. Please refer to [4], [5] for complete details of WCA.

TABLE I
Simulated Annealing Algorithm Notations

Notation	:	Meaning
N	:	total number of nodes
s	:	a random clusterhead set (dominant set)
$V(s)$	:	set of all possible dominant sets
$F(s)$	:	objective function
T_0	:	initial temp. of stepped geometric decrease
α	:	a constant to decrease temperature T
L	:	# of steps to run with unchanged temp.
ST	:	max. steps algo. runs without much change
ϵ	:	change in the objective function

A. Applying Simulated Annealing

In this section, we propose an optimized version of WCA by the use of simulated annealing algorithm (WCA_SA). Our goal is to optimize WCA such that the clusterhead set (dominant set) is minimized while load in the network is evenly balanced and each clusterhead node serves the maximum number of nodes. The maximum number of nodes that a clusterhead can serve which is defined as clusterhead's degree is configurable, meaning that, if a different network topology is presented, the existing topology can be adapted to the new scenario. In order to have a smaller number of clusterheads, each clusterhead must serve the maximum possible nodes within their clusters.

The terms used in our simulated annealing algorithm are shown in Table I. Furthermore, $F(s)$ gives the fitness value of dominant set s and defined as follows:

$$F(s) = \sum_{i=1} |s| W_{v_i}$$

1) WCA_SA Algorithm: Initially, we choose a solution s at random from neighborhood $V(s)$, then we compute the objective function $F(s)$ of solution s. Solution s and $F(s)$ will become the best solution and fitness value ever met during the course of the algorithm. Next, we choose any solution s' at random from $V(s)$; if $F(s, s') = F(s') - F(s) \leq 0$, which means if the fitness value of the next solution (solution s') that is generated by simulated annealing is less than or equal to the current best solution so far, then s' becomes the current solution; otherwise we decide with a certain probability whether s remains or s' becomes the current solution. However, in the case where $\Delta F(s, s') > 0$, the probability $(l, \Delta F)$ of moving s' as current solution decreases when $\Delta F(s, s')$

TABLE II
PARAMETERS USED

T_0	α	L	ST	ϵ
50	0.9	100	10	0.1

increases as well as with time. This strategy allows for escaping from the local minima, at least for small values of l, and tends to a classical descent algorithm when time elapses. In general, the probability $p(l, \Delta F)$ is defined by the following expression:

$$p(l, \Delta F) = exp(-l/T \times \Delta F(s, s'))$$

where T is called "temperature" by analogy with physical systems. T is a function of l and the main degree of freedom. There are several rather complex approaches to handle this, such as "cooling strategy", "random number" and so on. However, we have used a simpler approach called the "stepped geometric decrease". The algorithm starts at a chosen temperature T_0, and runs for L steps with unchanged temperature; then temperature is decreased at a constant factor $\alpha < 1$: $\alpha T_0 = T_1$; after L steps with temperature T_1, temperature is decreased to $T_2 = \alpha T_1$ and so on. The algorithm stops after a certain number of steps (ST) if there is no significant change in the objective function.

In the algorithm we have used the following five constant parameters: T_0, α, L, ST, and ϵ as described in Table I. In our implementation we choose the values for these constants as shown in Table II. These parameters are configurable from implementation meaning that the parameter values can be adjusted to serve a particular application.

2) Select_Random_Dominant_set: The goal of this sub-function is to select a random dominant set from a network of N nodes. A node ID starting from 1 to n uniquely identifies each node within a given network of N nodes. Each node in the network is either a clusterhead or a member of a clusterhead (neighbor). A node is a neighbor of a clusterhead if the distance between the node and its clusterhead is less than or equal to the transmission range. The Select_Random_Dominant_Set function starts by selecting a random node from the network of N nodes. If the node is neither clusterhead nor neighbor of other clusterhead, and its node degree is less than or equal to MAX_DEGREE, it becomes a clusterhead. It computes the weight (W_v) for this clusterhead and adds it into dominant set. This routine is repeated until every node in the network is either clusterhead or neighbor of a clusterhead.

3) Compute_Object_Function: This sub-function is mainly used to calculate the fitness value of dominant set. The fitness value of dominant set is calculated by summing up all the clusterhead's weight in the dominant set.

```
WCA_SA()
Begin
    Select_Random_Dominant_Set s
    Compute_Object_Function F(s)
    Replace s* by s And F* by F(s)
    While TRUE do
      Repeat L times
        Select_Random_Dominant_Set s
        Compute_Object_Function F(s')
        If F(s, s') ≤ 0 {
          s' becomes current solution s
          Compute_Object_Function F(s)
          If F(s) is less than F*
            Replace s* by s And F* by F(s)
        }
        Else {
          Generate a random number in [0,1]
          If random number ≤ p(1, ΔF) {
            s' becomes current solution s
            Compute_Object_Function F(s)
          }
      }
      Replace T by a * T(a < 1)
      If F is decreased by less than ε percent for
      a fixed # ST of consecutive series of L steps
        FALSE
    }
End


Compute_Object_Function()
Begin
   FitnessValue = 0
   For each ClusterHead in DominantSet
     FitnessValue = FitnessValue + W_v of ClusterHead
   Return FitnessValue
End
```

```
Select_Random_Dominant_Set()
Begin
    DominantSet = empty
    While there are remaining nodes
    do
        If (node is not a ClusterHead )
           and ( node is not a member of ClusterHead )
           and ( Degree of node ≤ MAX_DEGREE )
       {
           Compute weight W_v for this node
           Add node to DominantSet
       }
   }
End
```

III. Simulation Study

We simulate a system of N nodes on a 100×100 grid. The nodes could move in all possible directions with displacement varying uniformly between 0 to a maximum value (*max_disp*), per unit time. In our simulation experiments, N was varied between 20 and 60, and the nodes moved randomly in all possible directions with a maximum displacement of 10 along each of the coordinates. Every time unit, the nodes move a distance that is uniformly distributed between 0 and *max_disp*. Thus, the maximum Euclidean displacement possible is $10\sqrt{2}$. In the original WCA, we assumed that each clusterhead can at most handle $\delta = 10$ nodes (ideal degree) in its cluster in terms of resource allocation. Due to the importance of keeping the node degree as close to the ideal as possible, the weight w_1 associated with Δ_v was chosen high. The next higher weight w_2 was given to D_v, which is the sum of distances. Mobility and battery power were given low weights. The values used for simulation were $w_1 = 0.7$, $w_2 = 0.2$, $w_3 = 0.05$ and $w_4 = 0.05$. Note that these values are arbitrary at this time and should be adjusted according to the system requirements. The same values for all weighing factors are used both in the original and the optimized WCA.

A. Performance Metrices

We compare the performance of WCA with three performance metrics: (i) the number of clusterheads, (ii) the number of reaffiliations, and (iii) load balancing factor (LBF). The number of clusterheads in the network defines the *dominant set*. The reaffiliation count is incremented when a node gets dissociated from its clusterhead and becomes a member of another cluster within the current dominant set. These parameters are studied for varying number of nodes (N) in the system and maximum displacement.

To quantitatively measure how well balanced the clusterheads are, we use a parameter called *load balancing factor* (LBF) as was given in [4], [5]. Therefore, it is not desirable to have any clusterhead overly loaded while some others are lightly loaded. The load handled by a clusterhead is essentially the number of nodes supported by it. A clusterhead, apart from supporting its members with the radio resources, also has to route messages for other nodes belonging to different clusters. At the same time, it is difficult to maintain a perfectly load balanced system at all times due to frequent detachment and attachment of the nodes from and to the clusterheads. As the load of a clusterhead can be represented by the cardinality of its cluster size, the variance of the cardinalities will signify the load distribution. We define the LBF as the inverse of the variance of the cardinality of the clusters. Thus,

$$LBF = \frac{n_c}{\sum_i (x_i - \mu)^2}$$

where n_c is the number of clusterheads, x_i is the cardinality of cluster i, and $\mu = \frac{N-n_c}{n_c}$, (N being the total number of nodes in the system) is the average number of neighbors of a clusterhead. Clearly, a higher value of LBF signifies a better load distribution and it tends to infinity for a perfectly balanced system.

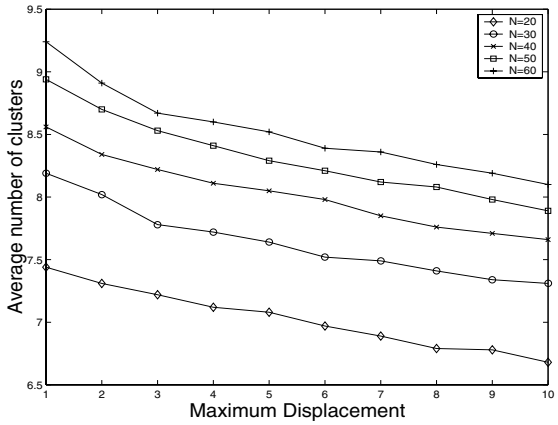

Fig. 1. Average number of clusters, *tx_range*=30

B. Experimental Results

Figures 1 and 2 show the average number of clusterheads with the varying *max_disp* for original WCA and optimized WCA respectively. We observe that the average number of clusterheads is almost the same for

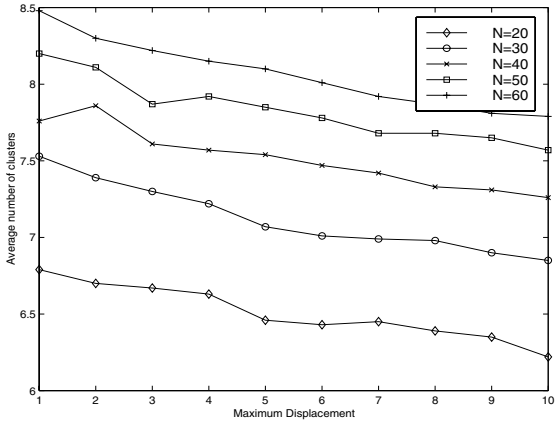

Fig. 2. Optimized WCA-Average number of clusters, *tx_range*=30

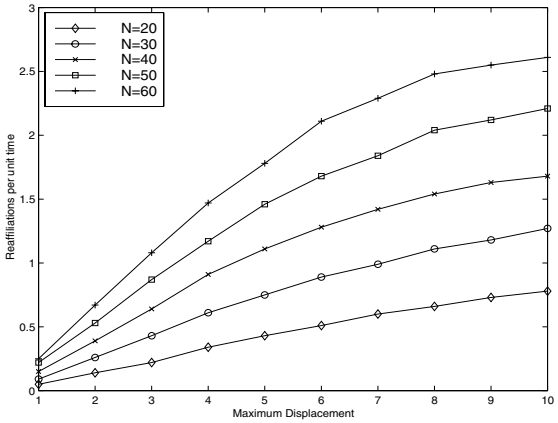

Fig. 4. Optimized WCA-Reaffiliations per unit time, *tx_range*=30

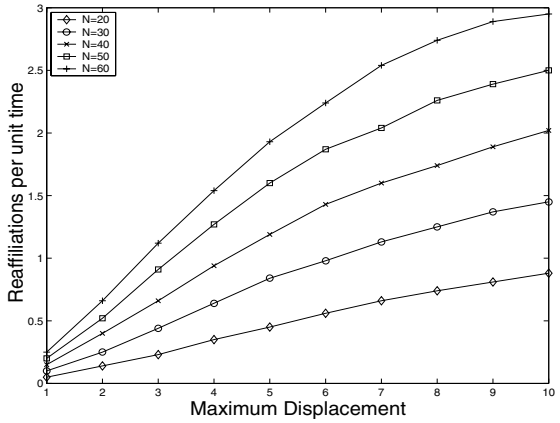

Fig. 3. Reaffiliations per unit time, *tx_range*=30

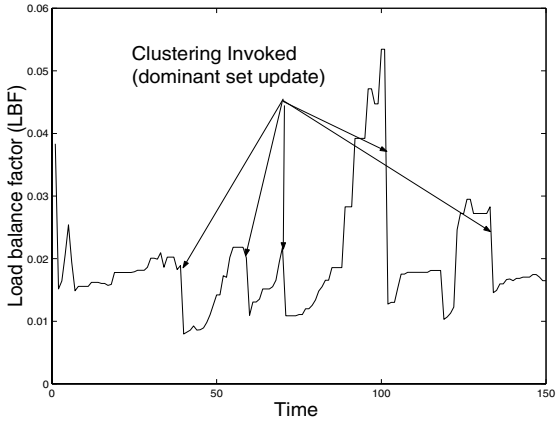

Fig. 5. Load distribution

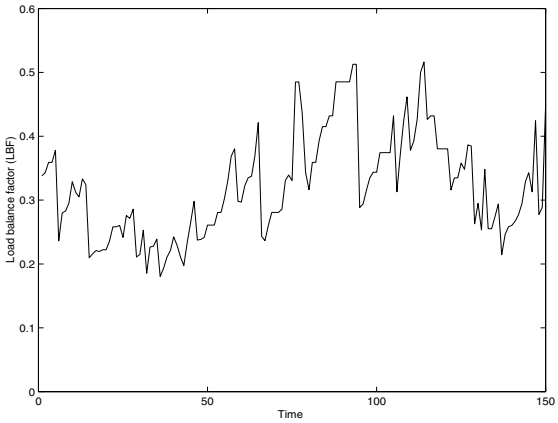

Fig. 6. Optimized WCA-Load distribution

different values of maximum displacement since it simply results in a different configuration but the cluster size remains the same. We observe that the optimized WCA yields lower number of clusters. Figures 3 and 4 show the reaffiliations per unit time with the varying *max_disp* for original WCA and optimized WCA respectively. As the displacement becomes larger, the nodes tend to move farther from their clusterhead, detaching themselves from the clusterhead and causing more reaffiliations per unit time. Figures 5 and 6 show how the load balancing factor (LBF) varies with time for original WCA and optimized WCA respectively. We observe that after every dominant set update, there is a *gradual* increase in the LBF. This gradual increase in LBF is due to the diffusion of the nodes among clusters. While the values of LBF has varied between 0 and 0.06 in Figure 5, it went up to 0.6 in Figure 6 indicating a ten times more balanced system.

IV. Conclusions

In this paper, we showed how simulated annealing can be applied to clustering techniques in mobile ad hoc networks. Weighted Clustering Algorithm (WCA) is

one such algorithm which can dynamically adapt itself with the ever changing topology of ad hoc networks. We have mapped the initial solution given by original WCA to simulated annealing algorithm in order to find the best possible solution from a set of all possible set of dominant sets. Data contained in the solution is used to compute the objective function of a particular solution according to the simulated annealing algorithm. The fitness value is computed in order to obtain the best solution by comparing each solution to the current best. This algorithm eventually finds the best solution without getting stuck in local minima by following the simulated annealing algorithm. Simulated annealing was applied to WCA to optimize its performance such that each clusterhead handles the maximum possible number of nodes in its cluster. The simulation results show that fewer clusterheads are obtained by applying simulated annealing to WCA than the results of the original WCA.

REFERENCES

[1] S. Basagni, I. Chlamtac, and A. Farago, "A Generalized Clustering Algorithm for Peer-to-Peer Networks", *Proceedings of Workshop on Algorithmic Aspects of Communication (satellite workshop of ICALP)*, Bologna, Italy, July 1997.

[2] B. Bollobas, *Random Graphs*, Academic Press, 1985.

[3] V. Cerny, "Thermodynamical Approach to the Traveling Salesman Problem", *Journal of Opt. Theory*, Vol. 45, 1985, pp. 41-51.

[4] M. Chatterjee, S.K. Das and D. Turgut, "An On-Demand Weighted Clustering Algorithm (WCA) for Ad hoc Networks", *Proceedings of IEEE GLOBECOM 2000*, San Francisco, November 2000, pp. 1697-1701.

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

[6] J.H. Kalivas, *Adaptation of Simulated Annealing to Chemical Optimisation Problems*, Elsevier, 1995.

[7] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi, "Optimization by Simulated Annealing", *Science 220*, 1983, pp. 671-680.

[8] P.J.M. van Laarhoven, *Theoretical and Computational Aspects of Simulated Annealing*, Centre for Mathematics and Computer Science, 1988.

[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines", *Journal of Chemistry and Physics*, Vol. 21, 1953, pp. 1087-1092.

[10] C.E. Perkins, Ed, *Ad Hoc Networking*, Addison Wesley, 2001.

[11] D.T. Pham and D. Karaboga *Intelligent Optimisation Techniques: Genetic Algorithm, Tabu Search, Simulated Annealing, and Neural Networks*, Springer, 2000.

[12] D. Turgut, S.K. Das, R. Elmasri, and B. Turgut, "Optimizing Clustering Algorithm in Mobile Ad hoc Networks Using Genetic Algorithmic Approach", *Proceedings of IEEE GLOBECOM 2002*, Taipei, Taiwan, November 17-21, 2002.