Large Wireless Ad Hoc Network Clustering with End-to-End QoS Constraints Using Multiobjective Particle Swarm Optimization

Nasser M. Alsaedi and Dionysios Kountanis

Abstract*—To scale for large ad hoc networks containing hundreds or even thousands of subnets, the networks must be hierarchically organized by partitioning them into clusters or domains. Many researchers have proposed partitioning of large networks into clusters, but this requires specifying cluster size bound. Graph Partitioning (GP) algorithms search for min-cut that balances the number of nodes in each partition. The limitations of GP algorithms are that they cannot be optimized for end-to-end performance requirements or take into account the characteristics of network topology. We propose a clustering algorithm based on the concept of Multiobjective Particle Swarm Optimization (MOPSO). Our algorithm minimizes the sum of the total hop count in each cluster and minimizes the edge-cut weight between clusters. Further, our algorithm provides quality of service (QoS) by taking into account end-to-end performance requirements. We find that minimizing the sum of the cluster hop count results in a balanced and bounded cluster size. For a given number of clusters, our clustering algorithm creates clusters with sizes closer to average size and avoids isolated nodes. *In our study, the terms domain, cluster, and partition are used interchangeably.

***Index Terms*—Ad-hoc network clustering, QoS, bounded cluster size, hop count, multiobjective optimization.**

I. Introduction

To build ad-hoc networks with hundreds or even thousands of nodes, the networks must be hierarchically organized to achieve scalable routing. Partitioning a network into smaller domains or clusters permits the organization of the network into hierarchical subnetworks. Hierarchical structures function by limiting the scope of route changes. They reduce the overall protocol overhead and the size of routing tables. Smaller domains permit routing QoS and other networking protocols to operate on fewer nodes, with inter-domain interaction through only a few links. Such division has two key benefits. First, it reduces the overall protocol overhead (e.g., routing overhead with n nodes reduces from $O(n^2)$ to $O(n \log n)$). Second, networking protocols can be tuned to more homogenous conditions if the domains are designed well [1]. For example, in the context of Open Shortest Path First (OSPF) [2], the partition should 1) create balanced non-backbone areas and 2) minimize the expected traffic among different areas so the backbone area is not overloaded (i.e., Area 0). The nodes in each area must be connected. Numerous clustering algorithms are introduced to partition the network into domains or clusters. Most clustering algorithms do not control the number of nodes in the clusters. Instead, they consider the diameter of clusters, convergence time, clusters stability, or energy consumption.

Preliminary research proposed algorithms for creating clusters where all nodes are one-hop away from the cluster head (CH) [3]-[5]. In such approaches, the average cluster size is limited. Other clustering algorithms require the cluster bound to be given. Ref. [6] introduced size-restricted distributed clustering for mobile ad-hoc networks. The cluster bound is provided to control the cluster size. Further, [7] proposed a clustering algorithm that provides QoS but the algorithm requires the cluster bound to be given. Clustering has been studied extensively in graph partitioning literature [8]-[10]. Graph partitioning methods search for min-cut to balance partition sizes. However, there are two disadvantages of graph partitioning methods. First, they do not take into account the characteristics of network. Second, most graph partitioning algorithms do not guarantee the connectivity of each partition. Ref. [11] introduced an optimization algorithm for large networks to generate connected partitions, but their algorithm still requires the partition bound.

We propose a multiobjective clustering algorithm to partition a network into connected partitions, considering the scalability property, and to provide end-to-end performance requirements. For a given number of clusters, our clustering algorithm creates clusters of sizes that are closer to the average size and avoids isolated nodes. The remainder of this paper is organized in the following manner: Section II provides a definition of the problem. Section III describes the Multiobjective Particle Swarm Optimization (MOPSO). Section IV discusses the effects of minimizing the total hop count on the network. Section V provides the objective functions of our clustering algorithm. Section VI describes our proposed multiobjective clustering algorithm. Section VII presents the experimental results, and Section VIII includes the conclusion and directions for future research.

II. Definition of the Problem

The network can be viewed as a weighted undirected graph $G = (V, E)$ consisting of a set of vertices V and a set of edges E, where each vertex $v \in V$ represents a network node and each edge $e \in E$ represents a communication link with an associated weight w_e. Our partitioning problem is to partition the graph G into k clusters (subgraphs) G_i, $i = 1, 2, \ldots, k$, that satisfy the following conditions:

Manuscript received February 12, 2014; revised April 18, 2014.

The authors are with Computer Science Department of Western Michigan University, USA (e-mail: nasser.m.alsaedi@wmich.edu, dionysios.kountanis@wmich.edu).

DOI: 10.7763/IJIEE.2014.V4.450

1) The sum of the total hop count in each cluster is minimized as much as possible.
2) The sum of the weights of all edge-cut is minimized.
3) Each cluster is connected, that is, there are no isolated nodes in any cluster.

The problem of partitioning the graph into *k* connected clusters such that every cluster minimizes the hop count between its vertices can be reduced to a graph partitioning problem [12]. Since the graph partition problem is an NP-complete problem, our problem is also NP-complete. We assume that the clustering algorithm has complete information regarding the network topology. We also assume availability of the traffic information in each node, node transition power, and the capacity and delay of each link. The link weight attribute determines the number and meaning of the link. For example, the link weights could cause link delay or the estimated link traffic to minimize the traffic between clusters.

Our clustering problem is formulated as an MOO problem and is expressed in the form of two objective functions. The first objective function generates clusters that minimize the sum of the total hop count within each cluster. The second objective function generates clusters that minimize the edge-cut weight between clusters.

III. Multiobjective Particle Swarm Optimization

Multiobjective optimization (MOO) [13] involves problems with two or more objectives. Evolutionary algorithms have been widely used to solve multiobjective problems. These algorithms are capable of identifying multiple solutions at a time instead of a single solution. Particle Swarm Optimization (PSO) is an evolutionary technique that is largely used for solving search and optimization problems [14]. PSO is a population-based metaheuristic inspired by the social behavior of birds within a flock. In a PSO algorithm, each potential solution to the problem is called *particle* and the population of solutions is called the *swarm*. The particles change their positions and move toward the best solutions found thus far.

PSO is expressed in the following manner: Let $\vec{x}_i$ denote the position of particle p_i, at time step *t*. The position of p_i is calculated by adding a velocity $\vec{v}_i(t)$ to the current position in the following manner:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \quad (2)$$

The velocity vector is defined in the following manner:

$$\vec{v}_i(t) = w.\vec{v}_i(t-1) + C_1.r_1.(\vec{x}_{p_i} - \vec{x}_i) + C_2.r_2.(\vec{x}_{g_i} - \vec{x}_i) \quad (3)$$

In Equation 3, $\vec{x}_{p_i}$ is the best solution that $\vec{x}_i$ has stored (called *pbest* of particle p_i), $\vec{x}_{g_i}$ is the best particle (called *gbest* of particle p_i, also known as the leader) that the entire swarm has obtained, *w* is the inertia weight of the particle and controls the trade-off between global and local experience, r_1 and r_2 are two uniformly distributed random numbers in the range (0, 1), and C_1 and C_2 are specific parameters that control the effect of the personal and global best particles. Many versions of the MOPSOs were introduced to solve multiobjective problems. Optimized MOPSO (OMOPSO) is the most salient algorithm from the MOPSOs studied in [15]. Further, OMOPSO has shown to be very vast in comparison with other state-of-the-art multiobjective metaheuristics [16]. The main feature of OMOPSO is that it incorporates the non-dominating sorting mechanism and crowding distance used in NSGA-II [17].

IV. Effect of Minimizing Cluster Hop Count

Minimizing hop count in the cluster implies minimizing the number of hops in every shortest path within the cluster. Hop count is the most commonly used routing metric in existing routing protocols for wireless ad hoc networks. Minimizing the hop count between all pairs within each cluster enriches the connectivity within the cluster and consequently achieves higher bandwidth utilization, reduces the latency within the clusters, and reduces energy consumption. In addition, minimizing the hop count between all pairs within each cluster actively attempts to minimize the edge cut between clusters. In our clustering algorithm, we observed that minimizing the sum of the total hop count in each cluster results in a balanced and bounded cluster size (i.e., the number of nodes in each cluster is the same within the smallest possible tolerance). For a given number of clusters, our clustering algorithm creates clusters of sizes closer to average size and avoids isolated nodes. This is because every node attempts to maximize its degree of connectivity within the cluster. In other words, every cluster attempts to maximize its size. Since the number of clusters is given, the resulting cluster sizes are as balanced as possible.

V. Objective Functions

Objective functions are employed to design clusters that minimize hop count and the edge-cut weight. In our experiments, we assign the same weight to all links, that is, the algorithm attempts to minimize the number of links between clusters. The following are the objective functions:

A. Minimize the Sum of Total Hop Count between All Pairs within Each Cluster

Let the number of clusters *C* equal *k*, and the hop count in cluster C_i is denoted by $H(C_i)$; then, the objective function is defined as

$$\text{Minimize } f_1 = \sum_{i=1}^{k} H(C_i) \quad (4)$$

B. Minimize the Edge-Cut Weights between Clusters

Let the number of clusters *C* equal *k*, and the edge-cut weight of cluster C_i is denoted by $W(C_i)$. The objective function that minimizes the edge-cut weights is defined as

$$\text{Minimize } f_2 = \sum_{i=1}^{k} W(C_i) \quad (5)$$

VI. The Multiobjective Clustering Algorithm

We propose a clustering algorithm based on OMOPSO to partition the network graph into *k* clusters. The graph vertices

V are associated with their coordinate's information. Each edge is assigned a weight. The shortest geodesic path is calculated between each pair. Each particle maintains vector C of length that equals the number of vertices in the graph, such that for every vertex $v \in V$, $C[v]$ indicates the cluster to which vertex v belongs (vertex membership). In our clustering algorithm, the particle is mapped to the k closest vertices in the graph and these vertices are considered cluster heads CHs. After identifying k CHs, each vertex is assigned to its nearest CH, forming k geodesic. The pseudocode of the OMOPSO-based clustering is included in Algorithm 1. Each particle initializes with k random locations representing the initial solution. After initializing the particles, the leaders are initialized and stored in the external archive; thereafter, the quality of leaders is calculated. In the main loop of the algorithm, for each particle, a leader is selected, the position is updated and, optionally, a mutation operator can be applied; thereafter, the particle partitions the network to k clusters. After forming k clusters, the particle's fitness is evaluated and then its corresponding p_{best} is updated. After each iteration, the set of leaders is updated and the quality of leaders is calculated. After the termination condition, the search solutions in the external are returned as the output of the algorithm. In each solution, the corresponding clustering guarantees that each vertex belongs to one cluster and each cluster is a connected component.

Algorithm 1: Pseudocode of the OMOPSO-based clustering algorithm

```
Input : Graph G with coordinates of vertices, number
        of cluster k
Calculate the Geodesic shortest paths between each
vertex pairs.
initialize swarm with random locations
initialize Leaders in the external archive
calculate Leaders quality
iteration = 0
while iteration < maxIteration do
    for each particle do
      select leader
      update Position    // Equations 2 and 3
      mutation           // optional
      map the particle to the closest vertices in the graph
      and consider these vertices as the cluster heads.
          for each vertex v in the graph do
              Assign v to the cluster that has nearest
              cluster head to v .
          end for
      evaluation        // evaluate objective functions
      update pbest
    end for
update Leaders in the external archive
calculate Leaders quality
iteration ++
end while
return solutions in the external archive as the outputs
```

VII. Experimental Results

We implemented the proposed algorithm in Java. For the PSO parameters, the swarm size is set to be 100 particles, the PSO iteration is set to be 400, the mutation operator is applied on 10% of the particles, and the size of the archive is set to be 100. For evaluating the performance of the proposed algorithm, we generate random graphs of different sizes where the nodes are randomly deployed according to a uniform distribution.

The first experiment evaluates the performance of our algorithm on a random graph G_1, where graph G_1 has 100 nodes in area of 120 × 120, the (x, y) coordinates of the nodes are generated randomly. The transmission range for each node equals 18 (i.e., each pair of nodes is connected if the distance between them is less than or equal to 18). All the edge weights are set to be one to minimize the number of edge-cut between clusters.

Table I presents the results of partitioning graph G_1 into 5 clusters under the objective, f_1 (Equation 4). The results include the cluster sizes, associated hop count, and number of edge-cut among the clusters. From Table I, we can see that the cluster sizes are bounded and closer to the average value (20). This implies the cluster sizes are almost balanced within the smallest possible tolerance. From the entirety of simulation runs, the obtained cluster sizes are 18, 19, 20, 21, and 22; obtained total hop counts are 950, 951, 952, 953, 954, and 955; and the obtained edge cut is between 31 and 43. Thus, it is evident that each cluster size has a corresponding hop count value associated with it. For example, when the cluster size is 20, the associated hop count is 190. From the results in Table I, it is evident that at the minimum value of the total hop count, 950, the cluster sizes are balanced. Moreover, it is also evident that if all the clusters have the same hop count, then all the clusters have the same size.

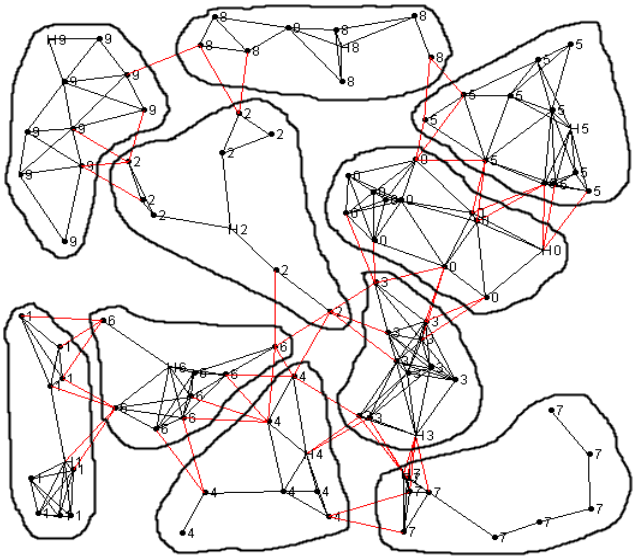

Fig. 1. Result of partitioning graph G_1 to 10 clusters under f_1 and f_2.

Table II presents the results of partitioning G_1 into 5 clusters under the two objectives, f_1 and f_2 (Equations 4 and 5). The first objective is minimizing the sum of the total hop count in each cluster. The second objective is minimizing the edge-cut weight. The obtained results correspond to the three solutions obtained from the final Pareto front. We can see that the results in Table II are a subset of the results in Table I. We observe that our algorithm satisfies the two objectives by choosing the non-dominating solutions that minimize the total sum of cluster hop count and edge-cut weight among clusters simultaneously. From the objectives' values, we can also observe that each objective value cannot be improved

without deteriorating the value of the second objective.

Table III presents the results of partitioning G_1 into 10 clusters under the two objectives, f_1 and f_2. The obtained results correspond to the three solutions obtained from the final Pareto front. From the results in Table III, it is evident that combining the two objectives bounds the cluster sizes, the generated clusters have average sizes closer to the average value (10), and the range of the cluster sizes is at a distance of (± 2) from the average size. Further, it is also evident from the objectives' values that each objective value cannot be improved without deteriorating the value of the second objective. Fig. 1 presents the first solution shown in Table III. Each cluster head is distinguished by a letter "H". From Fig. 1, we can see that each cluster is a connected component, the cluster sizes are between 8 and 12, and the edge-cut between clusters equals 60.

The second experiment evaluates the performance of our algorithm on random graphs of different sizes generated in the same area, 120 × 120, such that each graph is a connected component. We varied the number of nodes from 100 to 500 to vary the density of nodes. The graphs are partitioned into 5 and 10 clusters under the two objectives f_1 and f_2. Fig. 2 presents the range of the cluster sizes for the generated graphs. From Fig. 2, we can see that when the number of clusters equals 5, the resulting range of the cluster sizes is at a distance of (± 2) from the average size. When the number of clusters equals 10 and the number of nodes is less than 300, the range of cluster sizes is at a distance of (± 3) from the average size. When the number of cluster equals 10 and the number of nodes is greater than 300, the average of the cluster head is at a distance of (± 5) from the average size. This is because as the number of clusters increase, the search space also increases and then affects the results of the approximation algorithm.

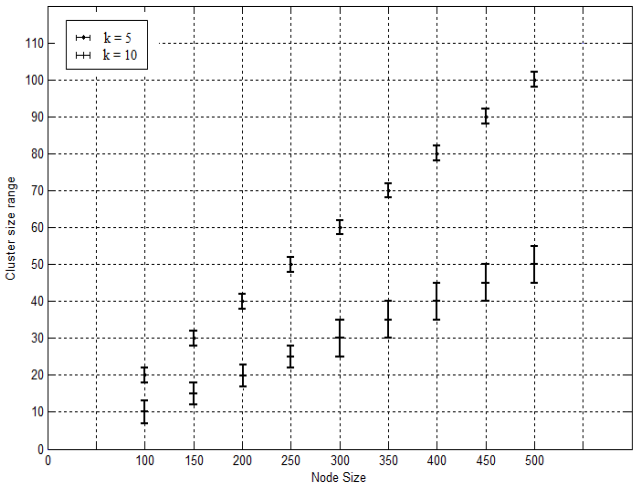

Fig. 2. Results of clustering of networks with different sizes.

TABLE I: RESULTS OF CLUSTERING GRAPH G_1 TO 5 CLUSTERS UNDER F_1

Cluster sizes	Clusters hop counts	Total hop count	Edge-Cut
(20 , 20 , 20 , 20 , 20)	(190 , 190 , 190 , 190 , 190)	950	34
(20 , 20 , 20 , 20 , 20)	(190 , 190 , 190 , 190 , 190)	950	38
(21 , 20 , 20 , 20 , 19)	(210 , 190 , 190 , 190 , 171)	951	33
(21 , 20 , 19 , 20 , 20)	(210 , 190 , 171 , 190 , 190)	951	43
(20 , 21 , 20 , 19 , 20)	(190 , 210 , 190 , 171 , 190)	951	34
(20 , 20 , 21 , 20 , 19)	(190 , 190 , 210 , 190 , 171)	951	39
(21 , 21 , 20 , 19 , 19)	(210 , 210 , 190 , 171 , 171)	952	31
(19 , 19 , 21 , 21 , 20)	(171 , 171 , 210 , 210 , 190)	952	34
(19 , 21 , 19 , 20 , 21)	(171 , 210 , 171 , 190 , 210)	952	42
(20 , 21 , 20 , 18 , 21)	(190 , 210 , 190 , 153 , 210)	953	39
(20 , 20 , 20 , 22 , 18)	(190 , 190 , 190 , 231 , 153)	954	42
(22 , 20 , 19 , 18 , 21)	(231 , 190 , 171 , 153 , 210)	955	34

TABLE II: RESULTS OF CLUSTERING GRAPH G_1 TO 5 CLUSTERS UNDER F_1 AND F_2

Cluster sizes	Clusters hop counts	Total hop count	Edge-Cut
(19 , 19 , 21 , 21 , 20)	(171 , 171 , 210 , 210 , 190)	952	31
(20 , 19 , 20 , 20 , 21)	(190 , 171 , 190 , 190 , 210)	951	32
(20 , 20 , 20 , 20 , 20)	(190 , 190 , 190 , 190 , 190)	950	34

TABLE III: RESULTS OF CLUSTERING GRAPH G_1 TO 10 CLUSTERS UNDER F_1 AND F_2

Cluster sizes	Total hop count	Edge-Cut
(12 , 10 , 9 , 11 , 8 , 12 , 9 , 10 , 9 , 10)	916	60
(10 , 11 , 10 , 10 , 11 , 8 , 9 , 11 , 11 , 9)	910	61
(9 , 10 , 11 , 10 , 11 , 10 , 10 , 9 , 9 , 11)	906	63

VIII. CONCLUSION AND FUTURE WORK

In this study, we investigated the problem of partitioning large ad hoc networks, thereby providing routing scalability and end-to-end QoS. We proposed a multiobjective clustering algorithm based on MOPSO to partition the network into connected partitions to solve the scalability problem and fulfill end-to-end performance requirements.

Our clustering algorithm does not consider the targeted size of cluster as a bound. It bounds the cluster sizes, and the resulting cluster sizes are closer to the average of the target size. The performance of the clustering algorithm is presented through its application to random graphs of multiple sizes where the nodes are randomly deployed according to a uniform distribution. In future research, we will attempt to find a heuristic that improves the results by balancing the cluster sizes as possible when the nodes are assigned to the CHs at each PSO iteration as well as identifying the appropriate number of clusters.

REFERENCES

[1] K. Manousakis, J. McAuley, R. Morera, and J. Baras, "Routing domain autoconfiguration for more efficient and rapidly deployable mobile networks," presented at 23rd Army Science Conference, Orlando, FL USA. Dec. 2-5, 2002.

[2] J. Moy, "OSPF Version 2," *IETF RFC*, no. 2328. April, 1998.

[3] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal of Selected Areas in Communications*, vol. 15, no. 7, pp. 1265-1275, 1997.

[4] S. Basagni, "Distributed clustering for ad hoc networks," in *Proc. International Symposium on Parallel Architectures, Algorithms and Networks*, Perth, Australia, 1999, pp. 310-315.

[5] M. Chatterjee, S. Das, and 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, pp. 193-204, 2002.

[6] G. Venkataraman, S. Emmanuel, and S. Thambipillai, "Size-restricted cluster formation and cluster maintenance technique for mobile ad hoc networks," *International Journal of Network Management*, vol. 17, no. 2, pp. 171-194, 2007.

[7] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *ACM Mobile Networks and Applications*, vol. 3, no. 1, pp. 101-119, 1998.

[8] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," in *Proc. the 1995 ACM/IEEE conference on supercomputing*, San Diego, CA, USA, 1995.

[9] H. D. Simon and S. H. Teng, "How good is recursive bisection?" *SIAM Journal of Scientific Computing*, vol. 18, no. 5, pp. 1436-1445, 1997.

[10] K. Andreev and H. Racke, "Balanced graph partitioning," *Theory of Computing Systems*, vol. 39, pp. 929-939, 2006.

[11] R. Krishnan, R. Ramanathan, and M. Steenstrup, "Optimization algorithms for large self-structuring networks," in *Proc. IEEE infocom 99*, Piscataway, NJ, 1999, vol. 1, no. 71, p. 8.

[12] M. R. Garey and D. S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-completeness," W. H. Freeman and Company, 1979.

[13] C. A. Coello, G. B. Lamont, and D. A. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-objective Problems*, New York: Springer, 2nd ed, 2007.

[14] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, San Francisco, CA: Morgan Kaufmann Publishers, 2001.

[15] J. J. Durillo, J. G. Nieto, A. J. Nebro, C. A. Coello, F. Luna, and E. Alba, "Multi-objective particle swarm optimizers: An experimental comparison," presented at 5th International Conference on Evolutionary Multi-Criterion Optimization (EMO'2009), New York, Springer, 2009.

[16] A. J. Nebro, J. J. Durillo, C. A. Coello, F. Luna, and E. Alba, "A study of convergence speed in multi-objective metaheuristics," in *Proc. the 10th International Conference on Parallel Problem Solving from Nature*, PPSN X, Dortmund, Germany, *Lecture Notes in Computer Science*, vol. 5199, pp. 763-772, 2008.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

Nasser M. Alsaedi received his master in computer science from Western Michigan University and currently a PhD student in Computer Science at Western Michigan University. His research interests are artificial intelligence, distributed systems, cloud computing.

Dionysios Kountanis is an associated professor in Computer Science Department at Western Michigan University. He received his master and PhD from University of Pennsylvania. His research interests are algorithm, artificial intelligence, and distributed systems.