

SACA: SCM-based Adaptive Clustering Algorithm

Yan Li¹, Snigdha Verma¹, Li Lao², Jun-Hong Cui¹

yan.li@uconn.edu, snigdha.verma@uconn.edu, llao@cs.ucla.edu, jcui@cse.uconn.edu

¹ Computer Science & Engineering Department, University of Connecticut, Storrs, CT 06269

² Computer Science Department, University of California, Los Angeles, CA 90095

Abstract

Network clustering is an important technique widely used in efficient hierarchical routing protocol design, network modelling and performance evaluation, etc. In this paper, we discuss the important clustering criteria, such as node connectivity, cluster diameter, number of orphan nodes. Our main contribution is a novel clustering algorithm SACA based on an accurate clustering measure called SCM. SACA adaptively forms clusters to incrementally improve the clustering quality, taking node connectivity into consideration. It can control the cluster size effectively and limit the number of orphan nodes. Our simulation study indicates that SACA is more accurate than MCL, a well accepted scalable and efficient clustering scheme, while requiring comparable running time for power law topologies and grid topologies, and significantly less running time for random topologies.

1 Introduction

Clustering is widely studied in various areas, such as biology, chemistry, linguistics, physics, and sociology. It is a very important method for data analysis. The basic goal of clustering is to group data into classes, with data in each class sharing certain similarity. In this paper, we investigate one interesting type of clustering: *graph clustering*, which is also called *network clustering* since a graph corresponds to a network topology in networking research. Network clustering provides a way to divide a network topology into groups/clusters such that nodes in the same cluster are highly connected and between clusters they are sparsely connected (a formal definition of network clustering is given in Section 3). If a network has many cluster sub-structures, we say it has clustering features.

The Internet can be treated as a huge network with clustering features at different levels: The Internet con-

sists of thousands of domains (or autonomous systems); and each domain is usually composed of many LANs (local area networks), with each LAN considered as a small densely connected network (or a cluster). In addition, recent work has shown that the AS (Autonomous System) level topology also has significant clustering features [2]. Grasping these clustering features is very useful in various network researches, such as hierarchical routing protocol design [6, 7, 9], network modelling [2, 4, 11], and performance evaluation, as arouses the need of good network clustering algorithms.

To design a good network clustering algorithm, probably the first question to ask is “what is a good network clustering algorithm?”. Generally speaking, the criteria of clustering heavily depends on applications. In networking research, however, a desired network clustering algorithm usually should have the following properties:

- Network nodes should be grouped into clusters based on their connectivity. Nodes in the same cluster should be highly connected, and least connected otherwise (according to the basic concept of network clustering).
- Cluster size (or cluster diameter) should be well controlled. In other words, the granularity of clustering should be adjustable.
- There should be a limited number of “orphan” nodes. An orphan node is defined as the one which is not grouped into a cluster with any other nodes.

We give a simple example to illustrate the above clustering properties in Fig. 1. In the shown topology, the cluster diameter threshold is set to 1. There are 4 clusters, and node 1 is left as an orphan node. This is a good clustering in the sense that it satisfies the above three properties.

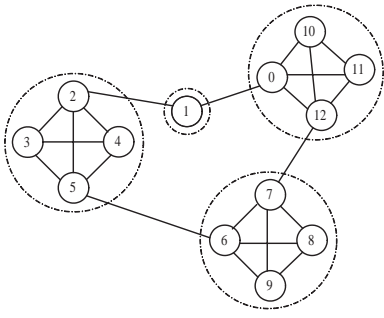


Figure 1. Clustering of a simple topology.
Cluster Diameter ≤ 1 .

In the literature, there has been considerable research in network clustering. Many clustering algorithms have been proposed, such as DDP [5], MCL [14], and CDC [12]. Although these algorithms address certain key problems, not all the three properties discussed above are satisfied. Moreover, the efficiency appears as a critical concern, which should also be a major goal of the clustering algorithm design. Therefore, the need is to design an efficient network clustering algorithm which satisfies all the properties.

With these problems in mind, in this paper, we present a novel network clustering algorithm, called **SCM-based Adaptive Clustering Algorithm (SACA)**. The main idea of our algorithm is to dynamically adjust cluster formation based on a clustering accuracy measure, called **SCM**(Scaled Coverage Measure), which is defined in [13]. SACA adaptively forms clusters to incrementally improve the clustering quality, taking node connectivity into consideration. It can control the cluster size effectively and limit the number of orphan nodes. We evaluate our SACA algorithm using various network topologies and compare it with MCL (which is well accepted as a scalable and efficient network clustering scheme). Our results show that SACA can form clusters with higher accuracy than MCL, while with comparable efficiency for power law topologies and grid topologies, and significantly higher efficiency for random topologies. Furthermore, cluster size and the number of orphan nodes are well controlled.

The rest of this paper is organized as follows. In Section 2, we review and discuss some existing clustering algorithms. In Section 3, we describe the network model and some notations used in this paper. Then in Section 4, we present our network clustering algorithm SACA. Finally, we evaluate the performance of SACA in Section 5, and conclude our paper in Section 6.

2 Related Work

Researchers have done significant work on network clustering and its applications [1, 5, 7, 8, 12, 14]. In this section, we review and discuss three typical clustering algorithms: DDP [5], MCL [14] and CDC [12].

2.1 DDP Algorithm

Dominating diametral path (DDP) algorithm is presented in [5]. The authors investigate a simplified version of the network clustering problem: the MINIMUM k -CLUSTERING problem.

The k -CLUSTERING problem is formulated as to find a partition of a given graph $G(V, E)$, such that the diameter of every cluster is at most k ($k \geq 1$). The MINIMUM k -CLUSTERING problem tries to find the smallest number of such clusters.

The authors give an efficient approximation algorithm for the MINIMUM k -CLUSTERING problem. However, this algorithm relies on the existence of a dominating diametral path (DDP). Since many graphs do not have such a path, the applicability of the DDP algorithm is quite limited. Moreover, this algorithm does not consider node connectivity. Therefore, the clustering is not necessarily accurate in term of connectivity.

2.2 MCL Algorithm

MCL [14] is a centralized cluster algorithm for graphs based on flow simulation. In this algorithm, an input graph G is mapped in a generic way onto a Markov matrix. Then the set of transition probabilities are iteratively recomputed via matrix expansion and inflation. An infinite sequence consisting of repeated alternation of expansion and inflation constitutes a new algebraic process called the Markov Cluster (MCL) process. The heuristic behind this algorithm is that a flow between sparsely connected dense regions evaporates after MCL process. Therefore, it is easy to detect dense regions in the original graph, and the dense regions are returned as clusters.

MCL algorithm is considered a scalable and efficient networking clustering scheme, which can achieve high clustering accuracy. However, some limitations still exist with MCL. Since clusters are formed by matrix operations, MCL is not time efficient when it processes large networks with a great number of nodes. Also, MCL is not suitable for dense networks in which nodes have high degrees. Finally, periodicity problem exists such that some graphs can not be clustered at all.

2.3 CDC Algorithm

The CDC (Connectivity-based Distributed Node Clustering) scheme is proposed by Ramaswamy et al. in [12]. It is a distributed and scalable algorithm for discovering connectivity-based clusters in peer-to-peer networks. The key idea of the CDC scheme is to simulate flows in the network. The concept of cluster head (i.e., originator in this scheme) is used to identify each cluster and the cluster diameter is well controlled by using TTL (time-to-live).

This scheme is scalable, and it can handle dynamic networks as well. However, its performance heavily relies on three thresholds, for which no optimized scheme exists. In addition, CDC does not have an effective way to eliminate orphan nodes. Therefore, good performance of the CDC scheme can not be guaranteed.

In short, there is no such clustering algorithms that satisfy all the three desired properties and can process all kinds of networks efficiently. In our work, we tackle this hard problem, and propose a more “aggressive” scheme based on SCM (Scaled Coverage Measure), the most practical clustering accuracy measure.

3 Network Model and Notations

In this section, we describe the network model and some important notations on network clustering.

3.1 Network Model

In this paper, we assume the network topology in consideration $G = (V, E)$ is a connected, undirected graph. V is the set of nodes, and E is the set of links. We denote $|V| = n$ and $|E| = m$. A partition $\mathcal{C} = \{C_1, C_2, \dots, C_l\}$ of V is called a *clustering* \mathcal{C} of graph G , and C_i s are called *clusters*. Obviously, $\bigcup_{i=1}^l C_i = V$. We call the maximum length of the shortest paths among all pairs of nodes in cluster C_i as its *size*, or *diameter*. If a cluster has a size of 0, i.e., it has only one node, this cluster is a trivial cluster, which is also referred to an *orphan node*.

A network clustering algorithm is to find a “good” partition \mathcal{C} of graph G with the goal of maximizing the clustering feature, i.e., the nodes in the same cluster should be highly connected and the nodes between clusters are least connected. To be concrete, a good network clustering algorithm should consider node connectivity, cluster size, and orphan nodes.

Node Connectivity Node connectivity is classified into intra-cluster connectivity and inter-cluster connec-

tivity. Assume node v_i is in cluster C_i , we define the **intra-cluster connectivity** of node v_i as the number of links between v_i and its neighbors in the same cluster C_i . Accordingly, the **inter-cluster connectivity** of node v_i is defined as the number of links between v_i and its neighbors which are not in cluster C_i . For instance, in Fig. 1, node 0’s intra-cluster connectivity is 3, while it’s inter-cluster connectivity is 1.

A good clustering algorithm should be capable of maximizing the intra-cluster connectivity and minimizing the inter-cluster connectivity of nodes. In addition, for any node v_i , the number of non-neighbor nodes in the same cluster should be minimized. For example, in Fig. 1, each node has 0 non-neighbor node in the same cluster.

Cluster Size Besides node connectivity, cluster size is also an important metric to consider. It may not be desirable that clusters are too expanded in most of the scenarios. The number of hops along the longest shortest path among all pairs of nodes in a cluster, i.e., cluster diameter, is usually used to reflect the cluster size (as is defined above). A threshold can be used to control the cluster diameter. Different thresholds may influence the cluster formation significantly.

Orphan Nodes In network clustering, it is possible that some nodes are left without joining any cluster (due to the limit of cluster diameter and the demand of maximizing clustering accuracy). These nodes are orphan nodes. In most scenarios, orphan nodes are not preferred and should be eliminated.

3.2 Scaled Coverage Measure (SCM)

How to measure the accuracy of clustering? This is probably the most important question to answer before the network clustering algorithm design. Considering the requirements of good clustering, we choose **Scaled Coverage Measure (SCM)** as the metric.

SCM is an intuitive and practical measure proposed by S. van Dongen in [13]. This measure is based on the idea that an accurate clustering should minimize the inter-cluster connectivity as well as the number of non-neighbor nodes in each cluster.

For graph G with clustering \mathcal{C} , given node v_i , we first have the following notations:

$0.5\text{cmNbr}(v_i)$ is the set of neighbors of v_i ;

$0.5\text{cmClust}(v_i, \mathcal{C})$ is the set of all nodes that are in the same cluster as node v_i (not including v_i);

$0.5\text{cmFalsePos}(v_i, \mathcal{C})$ is the set of all nodes which are in the same cluster as v_i , but are not neighbors of v_i ;

$0.5\text{cmFalseNeg}(v_i, \mathcal{C})$ is the set of all neighbors of v_i but are not included in the same cluster as v_i .

Then the *Scaled Coverage Measure*, $SCM(v_i, \mathcal{C})$, of

node v_i is defined as:

$$1 - \frac{|FalsePos(v_i, \mathcal{C})| + |FalseNeg(v_i, \mathcal{C})|}{|Nbr(v_i) \cup Clust(v_i, \mathcal{C})|}. \quad (1)$$

The SCM of a graph G , $SCM(G)$, is defined as the average of the SCM values of all the nodes, that is, $SCM(G) = (\sum_{v_i} SCM(v_i, \mathcal{C})/n)$. It lies in $[0, 1]$.

SCM can well reflect the clustering accuracy: the higher the SCM, the less the inter-cluster connectivity and the smaller the number of non-neighbor nodes (as translates into higher intra-cluster connectivity). In addition, it can be seen that for the graphs which contain only fully connected clusters/subgraphs and there are no links across the clusters, so its SCM can reach 1. This is consistent with our intuition, demonstrating the power of SCM. Furthermore, the SCM of an orphan node is 0, which matches our goal of minimizing the number of orphan nodes.

Therefore, the network clustering problem can be simplified as partitioning a network topology such that its SCM can be maximized. Our proposed algorithm, SACA, is exactly based on SCM, adaptively forming clusters in an aggressive manner.

4 The Algorithm

In this section, we present our algorithm SACA (SCM-based Adaptive Clustering Algorithm). First we give a short overview, then we explain the algorithm in details. We also discuss some related issues.

4.1 Overview

Given the considerations in the previous sections, it seems critical to develop an efficient network clustering algorithm which can effectively control the cluster size, minimize the node inter-cluster connectivity while maximizing the intra-cluster connectivity, and end with a minimal number of orphan nodes. SACA achieves these goals by adaptively adjusting the cluster formation in order to maximize the SCM, while controlling the cluster diameter.

The basic procedure of SACA is as follows: For a graph G , SACA first initializes all the nodes as orphan nodes. Then each ‘‘clustered’’ node tries to merge with its ‘‘non-clustered’’ neighbors based on the gain of SCM (that is, if the merge attempt increases the SCM of graph G , the merge action is conducted). A ‘‘clustered’’ node can also move from one cluster to another cluster, and the moving action is also justified by the gain of SCM.

In the following, we explain each of the key steps of SACA in details.

4.2 Initialization

The input of SACA is a network topology and a cluster diameter threshold. SACA first initializes every node as an orphan node. For convenience, the Scaled Coverage Measure of node v_i can be denoted as:

$$SCM(v_i) = 1 - \frac{b_{v_i}}{a_{v_i}}, \quad (2)$$

Thus,

$$b_{v_i} = |FalsePos(v_i, \mathcal{C})| + |FalseNeg(v_i, \mathcal{C})|,$$

and

$$a_{v_i} = |Nbr(v_i) \cup Clust(v_i, \mathcal{C})|.$$

At the beginning, $b_{v_i} = a_{v_i} = deg(v_i)$, where $deg(v_i)$ denotes the degree of node v_i .

4.3 Node Merging

After initialization, nodes should be merged together as clusters based on SCM. SACA always merges an orphan node (or ‘‘un-clustered’’ node) with such neighbors that the SCM of the whole network can be maximized after this step. However, the SCM of the whole topology is not easy to compute. Thus, SACA computes the gain in SCM and merges the orphan nodes with the neighbors which result in a positive and the largest gain in SCM.

The computation for the gain in SCM is quite simple. For an orphan node v_i , if it is merged with a cluster C_0 , the SCM values of the following nodes might be changed: node v_i , the neighbor nodes of v_i in cluster C_0 , the non-neighbor nodes of v_i in cluster C_0 .

The change in node v_i 's SCM is:

$$\Delta SCM(v_i) = \frac{n_0}{deg(v_i) + m_0 - n_0}, \quad (3)$$

where m_0 is the number of nodes in cluster C_0 , and n_0 is the number of neighbor nodes of v_i in cluster C_0 .

Accordingly, if a node v_j is a neighbor of node v_i in cluster C_0 , the change in its SCM is:

$$\Delta SCM(v_j) = \frac{1}{a_{v_j}}, \quad (4)$$

where $v_j \in neighbors_{v_i}$.

On the other hand, if v_j is a non-neighbor node of v_i in cluster C_0 , the change in its SCM is given by:

$$\Delta SCM(v_j) = \frac{b_{v_j}}{a_{v_j}} - \frac{b_{v_j} + 1}{a_{v_j} + 1}, \quad (5)$$

where $v_j \in non_neighbors_{v_i}$.

For other nodes, the SCM values are not changed. Therefore, the gain in SCM of the whole network G if node v_i joins cluster C_0 is:

$$\Delta SCM(G) = \frac{\Delta SCM(v_i) + \sum_{v_j \in C_0} \Delta SCM(v_j)}{n} \quad (6)$$

where n is the number of nodes in the network G .

To simplify the computation, we ignore the divisor n , which is constant for a given topology. If $\Delta SCM(G) > 0$, the overall SCM of graph G will be increased, thus node v_i will join C_0 . In this way, SACA merges nodes one by one based on the gain in SCM.

Cluster Diameter Control When a node attempts to join a cluster, SACA also checks the diameter of the target cluster. If the cluster size exceeds the given threshold after the node joins, this merge action will be cancelled, and the node will check other candidate clusters to join.

4.4 Node Moving

As is mentioned above, orphan nodes join clusters based on the gain in SCM. However, it is not necessary that a node which has been grouped into a cluster should be in that cluster forever. The reason is that nodes are processed one by one, and the order of nodes being processed may influence the cluster formation significantly. For example, in Fig. 2, if SACA processes the nodes in the order of (1, 2, 0, 3, 4, 5), then we get two clusters as shown in Fig. 2. a, with node 0 clustered with nodes 1 and 2. While if the processing order is (1, 2, 3, 0, 4, 5), we may obtain a different set of clusters, which are illustrated in Fig. 2. b, where node 0 is grouped with nodes 3, 4, 5. In this example, we assume the cluster diameter threshold is 2.

To solve this problem, SACA allows the movement of nodes from one cluster to another. Quite similar to the operation of merging an orphan node into an existing cluster, the moving procedure is also based on the gain in SCM. Whenever an orphan node v_i joins a cluster, its neighbor node, v_j , which is not in the same cluster as node v_i , will be checked to see if it should move to v_i 's cluster. Basically, SACA will compute the gain in $SCM(G)$ if v_j leaves its own cluster, Δ_{leave} , and then compute the gain in $SCM(G)$ under the assumption that v_j joins v_i 's cluster, Δ_{join} . The overall gain in $SCM(G)$, therefore, is computed as:

$$\Delta SCM(G) = \Delta_{join} + \Delta_{leave} \quad (7)$$

If $\Delta SCM(G) > 0$, node v_j should move from the original cluster to v_i 's cluster.

Since v_j 's movement may affect its neighbors which are not in the same cluster as v_j , these neighbor nodes

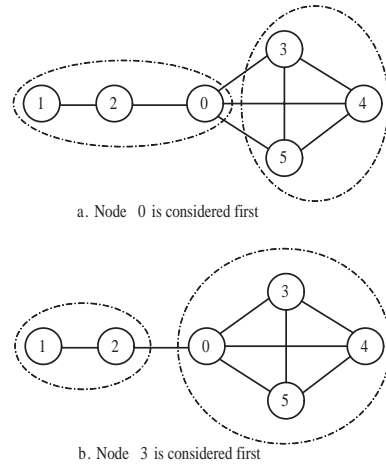


Figure 2. Different clusters under different processing order

should also be checked to determine if they should move.

After each merging and moving operations, every involved node should update its b and a (defined in Equation 2) accordingly. The merging and moving procedures are carried out until there is no orphan node or all the orphan nodes can not join any cluster due to the constraint of cluster diameter threshold.

4.5 A Simple Clustering Example

To illustrate the SCA algorithm, we show a simple example in Fig. 3. At the beginning, all the 8 nodes in the network are orphan nodes. Assume we start from node 0. Then it chooses its neighbor node 1 to merge. Now, nodes 0 and 1 form a cluster shown in Fig. 3. a. When SACA scans the orphan node 2, due to the constraint of cluster diameter (in this example, we assume the diameter threshold is 1), node 2 can not join the cluster with nodes 0 and 1. Thus, it selects the best candidate (if there is a tie, we randomly select one candidate) node 3 to merge. So far, we have two clusters shown in Fig. 3. b. However, since node 0 is a neighbor of node 2, which just joined a new cluster, SACA needs to check the possibility of moving node 0. By computing the SCM gain of this move, it is judged that node 0 should move to the cluster with nodes 2 and 3. After the moving action, we only have one bigger cluster with nodes 0, 2, and 3 (illustrated in Fig. 3. c). Now, node 1 becomes an orphan node. Thus, here we can see node moving might result in new orphan nodes, which will be considered in later SACA scanning iterations. Following this way, we get

the final clustering which is depicted in Fig. 3. d. In this final state, any join action of the orphan node 1 will not increase the SCM of the network under the constraint of cluster size 1.

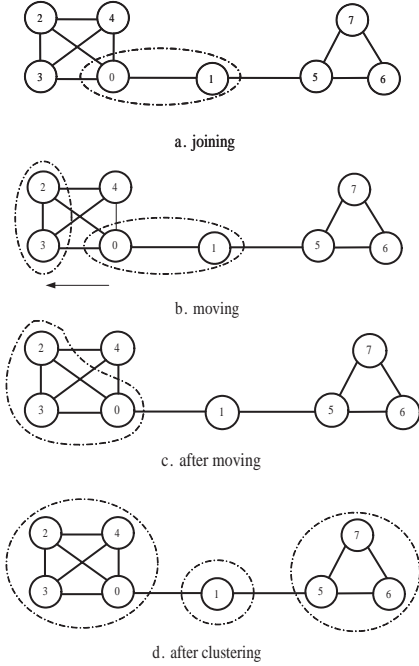


Figure 3. An example of SACA algorithm

4.6 SACA Algorithm

We summarize SACA in Algorithm 1. All the nodes are first initialized as orphan nodes. Then the SACA algorithm starts an adaptive procedure. In each adaptive iteration, SACA passes through the whole node set. For each orphan node v_i , SACA chooses a best neighbor cluster for v_i to join, based on SCM and cluster size constraint at the same time. v_i 's join might cause the move of its other neighbors, which may again trigger further node move. This is a recursive procedure (shown in Algorithm 2). The algorithm stops when there are no orphan nodes or there are no move or join actions. Following this procedure, we can see that the merging (or joining) and moving operations of SACA help to optimize cluster formation in such a way that the overall SCM is maximized.

4.7 Discussions

Distributed Implementation Although we present SACA in a centralized manner, it is easy to implement it in a distributed fashion. If we check the

Algorithm 1 SACA($G(V, E), k$)

```

1: //cluster diameter threshold is  $k$ ,
2: for all  $v_i \in V$  do
3:    $a_{v_i} = b_{v_i} = deg(v_i)$ 
4:   //Initialize  $v_i$  as an orphan node
5: end for
6: while there is orphan node AND there is join or
   move action do
7:   for all  $v_i \in V$  do
8:     //pass all the nodes in  $G$ 
9:     if  $v_i$  is an orphan node then
10:      //start a join procedure
11:      for all node  $v_j \in neighbors_{v_i}$  do
12:         $C =$  cluster of  $v_j$ 
13:        if  $diameter(C) \leq k$  when  $v_i$  joins then
14:          compute  $\Delta SCM(G)$  as if  $v_i$  joins  $C$ 
15:        end if
16:      end for
17:      for all possible join actions, find a cluster  $C_0$ 
        with the largest positive  $\Delta SCM(G)$ 
18:      node  $v_i$  joins cluster  $C_0$ 
19:      call function  $MOVE(v_i, C_0)$ 
20:      //start a move procedure
21:    end if
22:  end for
23: end while

```

Algorithm 2 MOVE(v_i, C_0)

```

1: for all node  $v_s \in neighbors_{v_i}$  do
2:   if  $v_s$  is not in cluster  $C_0$  then
3:     compute  $\Delta SCM(G)$ 
4:     if  $\Delta SCM(G) > 0$  AND diameter of  $C_0 \leq k$ 
       then
5:       move  $v_s$  to cluster  $C_0$ 
6:       call function  $MOVE(v_s, C_0)$ 
7:     end if
8:   end if
9: end for

```

SACA algorithm carefully, we will find that all the operations (SCM gain computation, initialization, join action, and leave action) are localized, and no global information is needed. Thus, in a real network, if each node is equipped to obtain its neighbor information and its cluster information (which can be easily done by some local information exchange), the network can be partitioned into clusters in a distributed way.

Dynamics Handling In peer-to-peer networks, nodes join and leave, and it is challenging to maintain meaningful clusters in such networks with high dynamics. Due to the inherent adaptive feature of our SACA algorithm, it can handle node dynamics easily. Node

join can naturally emulate the node merging procedure. For node leave, we can check the affected nodes and trigger the node moving procedure when necessary.

5 Performance Evaluation

In this section, we conduct simulations to evaluate the performance of SACA, comparing it with MCL, a well-accepted network clustering algorithm.

5.1 Experiment Settings

We use topologies generated from three sources: GT-ITM, BRITE, and Grid. The GT-ITM [3] topology generator produces random topologies, and the BRITE [10] topology generator provides power law topologies. Grid topologies, even though less realistic than the other types of topologies, is used to test the applicability of our clustering algorithm to various topologies.

We implement SACA in C++ and use the public domain software of MCL [13]. The performance of MCL heavily depends on a configurable parameter, which controls the clustering granularity. We measure the performance of MCL with all the possible values for this parameter, and choose the best result to compare with that of our algorithm.

We use two metrics: clustering accuracy and efficiency. We compute the clustering accuracy using SCM, and measure the efficiency in terms of the time taken by each algorithm to generate clusters. In addition, we also study the influence of diameter threshold on the number of clusters and the clustering accuracy.

5.2 Clustering Accuracy

In this set of experiments we compare the accuracy of SACA with the MCL.

For the topologies generated by GT-ITM, the number of nodes is varied from 15 to 8000. The average degree is about 10 for the topologies which have more than 100 nodes, and 3 to 5 for the rest.

Fig. 4 and 5 display the clustering accuracy of SACA and MCL. These results show that SACA can generate more accurate clusters for random topologies. It gives an improvement of about 37% over MCL for small topologies (200 to 1000 nodes). For larger topologies (1000 to 8000 nodes), the improvement is even higher. Furthermore, this improvement shows an increasing trend as the topology size is further enlarged. It is clear that the adaptive adjustment of clusters in SACA, which always tends to maximize the clustering accuracy, can result in better performance even for

large topologies while MCL’s performance deteriorates significantly.

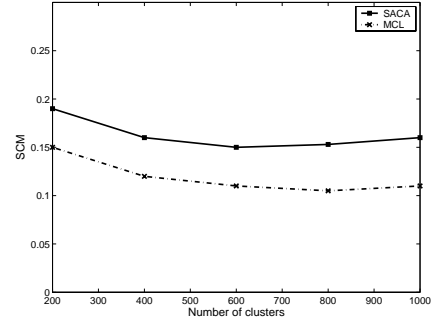


Figure 4. Clustering accuracy of random topologies having 200 to 1000 nodes.

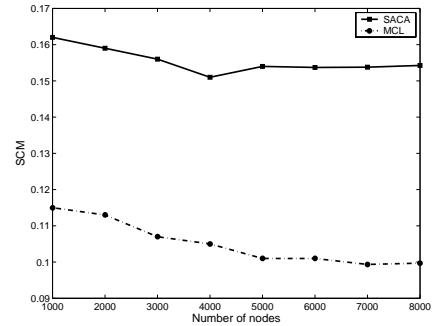


Figure 5. Clustering accuracy of random topologies having 1000 to 8000 nodes.

We also compare the accuracy of the two algorithms for power law topologies generated by BRITE. For these topologies, we vary the number of nodes from 1000 to 7000. Fig. 6 shows the clustering accuracy of the two algorithms. The results indicate that SACA has about 10% improvement over MCL.

Grid topologies are quite different from the above two in that each node can communicate with its immediate horizontal and vertical neighbors. Since links are homogeneous in grid topologies, MCL’s flow simulation is less likely to work well. We compare the clustering accuracy of these two algorithms in grid topologies and plot the results in Fig. 7. It can be observed from the figure that SACA achieves much better performance than MCL, since SACA can handle grid topologies easily by partitioning the whole topology evenly, while MCL gets either very small or very large clusters and results in a low accuracy.

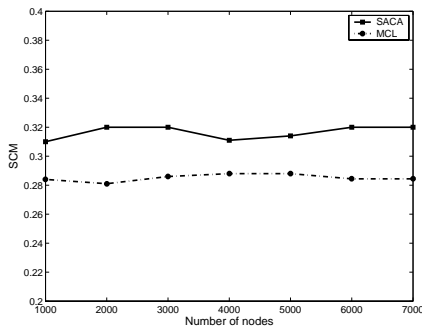


Figure 6. Clustering accuracy of power law topologies having 1000 to 7000 nodes.

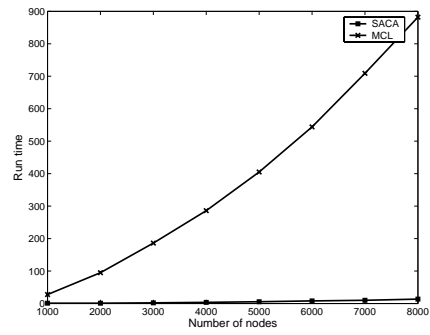


Figure 8. Run time of SACA and MCL for random topologies having 1000 to 8000 nodes.

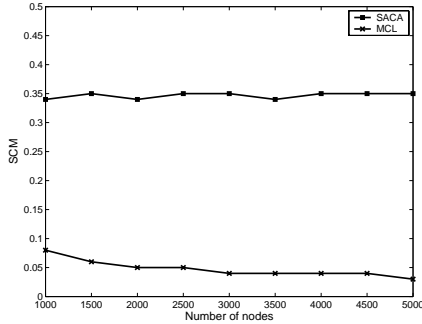


Figure 7. Clustering accuracy of grid topologies having 1000 to 5000 nodes.

of clusters is decreasing with the increase of diameters, which is consistent with our intuition. We also observe similar behavior for other topologies.

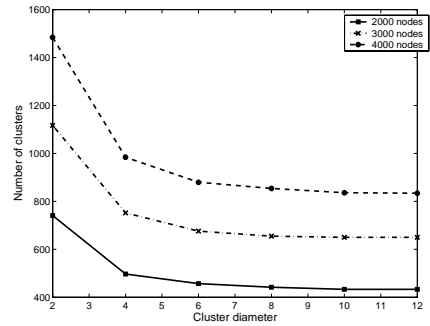


Figure 9. Influence of the diameter threshold on the number of clusters for power law topologies having 2000 to 4000 nodes.

5.3 Efficiency

In these experiments, we compare the run time of SACA with MCL to test the efficiency of both the clustering schemes. Fig. 8 indicates that there is a significant difference between the run time of both algorithms for random topologies. Moreover, this difference tends to increase as the topology gets larger. As for power law and grid topologies, MCL yields a slightly better efficiency than SACA: MCL is no more than several seconds faster than SACA in most cases. However, we want to emphasize that, in these two cases, SACA outperforms MCL by a large margin while only spending a comparable amount of time.

5.4 Effect of Diameter

Since SACA controls the diameter of clusters, it is of interest to study the influence of diameter on the clustering. Fig. 9 shows the change in the number of clusters with different diameter thresholds for power law topologies. The figure indicates that the number

While the number of clusters decreases with the increase in diameter, the clustering accuracy is not necessarily affected. Fig. 10 shows the results of SCM of power law topologies when the diameter threshold varies. Intuitively, there are two reasons that an orphan node cannot merge with other clusters: the diameter constraint is violated, or there is no improvement in SCM. As a result, when the diameter threshold is low, there are many orphan nodes due to the first reason. As the threshold is raised, the number of orphan nodes will decrease, and thus the clustering accuracy will improve. However, when the threshold reaches beyond a certain point, it does not have much impact on the number of orphan nodes and the clustering accuracy. This trend is clearly shown in the figure.

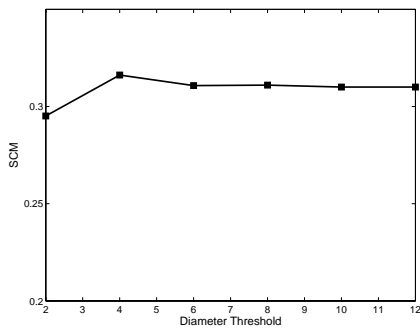


Figure 10. Influence of the diameter threshold on the cluster accuracy for power law topology with 2000 nodes.

5.5 Summary

In summary, our simulation study demonstrates the promising performance of SACA. We find that: (1) In all the topologies tested, SACA consistently achieves significantly higher accuracy than MCL in terms of SCM; (2) In random topologies, SACA is much faster than MCL, especially as topology size increases. In power law and grid topologies, SACA has comparable efficiency with MCL; (3) As the cluster diameter threshold increases, there are fewer but larger clusters. On the other hand, the clustering accuracy is only affected slightly by the diameter threshold.

6 Conclusions

We have presented SACA (SCM-based Adaptive Clustering Algorithm), an accurate and efficient clustering scheme for various networks. We first identify a comprehensive set of practical properties for good clustering algorithms. The basis of SACA, SCM (Scaled Coverage Measure), is then verified as an effective and accurate measure for clustering. In SACA, clusters are formed in an adaptive way: nodes join and leave their clusters only if the SCM value of the whole network can be improved. The algorithm design is proved to be simple and it only requires localized operations. Experiment results show that SACA is very efficient and accurate even for large topologies. Finally, SACA has a good control of the diameter while limiting the number of orphan nodes.

References

[1] A. Barbu and S.-C. Zhu. Graph partition by swendsen-wang cuts. In *Ninth IEEE International Conference*

on Computer Vision, Nice, France, volume 1, pages 320–329, 14–17 October 2003.

[2] T. Bu and D. F. Towsley. On distinguishing between internet power law topology generators. In *Proceedings of INFOCOM*, 2002.

[3] K. Calvert and E. Zegura. Gt-itm: Georgia tech internet network topology models. <http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz>, 1996.

[4] J.-H. Cui, M. Faloutsos, D. Maggiorini, M. Gerla, and K. Boussetta. Measuring and modelling the group membership in the internet. In *Proceedings of ACM-SIGCOMM/USENIX Internet Measurement Conference (IMC2003)*, Miami, Florida, pages 65–67, October 27–29, 2003.

[5] J. S. Deogun, D. Kratsch, and G. Steiner. An approximation algorithm for clustering graphs with dominating diametral path. volume 61, pages 121–127, Amsterdam, The Netherlands, The Netherlands, 1997. Elsevier North-Holland, Inc.

[6] D. Estrin, Y. Rekhter, and S. Hotz. Scalable inter-domain routing architecture. In *Proceedings of SIGCOMM*, pages 40–52, 1992.

[7] Y. Fernandess and D. Malkhi. K-clustering in wireless ad hoc networks. *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 31–37, 2002.

[8] C. Gkantsidis, M. Mihail, , and E. Zegura. Spectral analysis of internet topologies. In *IEEE INFOCOM*, October 27–29, 2003.

[9] A. McDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. In *IEEE Journal on Selected Areas in Communication*, volume 17, page 8, August 1999.

[10] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: Universal topology generation from a user’s perspective. In *Proceedings of International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '01)*, October 2001.

[11] A. Medina, I. Matta, and J. Byers. On the origin of power laws in internet topologies. In *SIGCOMM Comput. Commun. Rev.*, volume 30, pages 18–28, New York, NY, USA, 2000. ACM Press.

[12] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis. Connectivity based node clustering in decentralized peer-to-peer networks. In *3rd International Conference on Peer-to-Peer Computing*, 2003.

[13] S. van Dongen. Performance criteria for graph clustering and markov cluster experiments. *Technical report*, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, 2000.

[14] S. van Dongen. A new cluster algorithm for graphs. *Technical Report INS-R0010*, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, May 2000.