Graph Clustering Using Distance-k Cliques

Jubin Edachery[1], Arunabha Sen[1] *, and Franz J. Brandenburg[2]

[1] Department of Computer Science
Arizona State University
Tempe 85287, AZ, USA
email: {jubin, Arunabha.Sen}@asu.edu
[2] Lehrstuhl für Informatik
Universität Passau
94030 Passau, Germany
email: brandenb@informatik.uni-passau.de

Abstract. Identifying the *natural clusters* of nodes in a graph and treating them as *supernodes* or *metanodes* for a higher level graph (or an abstract graph) is a technique used for the reduction of visual complexity of graphs with a large number of nodes. In this paper we report on the implementation of a clustering algorithm based on the idea of *distance-k cliques*, a generalization of the idea of the *cliques* in graphs. The performance of the clustering algorithm on some large graphs obtained from the archives of Bell Laboratories is presented.

1 Introduction

Visualization tools can be of tremendous help to network planners and administrators for management and control of large networks. For this purpose, a hierarchical view of the network is most appropriate. This provides the network administrators at different levels the ability to view their realm of the network at an appropriate level of detail without being overwhelmed by unnecessary and unimportant minutiae. This hierarchical view of the networks can be provided in graphs by grouping the nodes into some *supernodes* or *metanodes* [3]. As a result, the higher level graph will have much fewer metanodes and thus the visual complexity will be significantly reduced.

Identifying the *natural clusters* of nodes in a graph and treating them as supernodes or metanodes for a higher level graph is a technique that can used for the reduction of visual complexity of graphs with a large number of nodes. An abstract graph is constructed in such a way that a node in the abstract graph represents a set of nodes of the original graph and the edges represent the relationship between these sets of nodes. Thus an abstract graph construction problem reduces to the problem of partitioning the node set of the original graph $G = (V, E)$, into a subset of nodes $V_1, V_2, ..., V_k$, such that $\cup_{i=1}^{k} V_i = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$.

The subsets V_is ($1 \leq i \leq k$) are known as the *clusters* of the graph $G = (V, E)$. One problem with this approach is that there is no consensus among the researchers

* Corresponding author, Telephone: 480-965-6153, Fax: 480-965-2751; This research in part was supported by a grant from the NATO Scientific and Environmental Affairs Division and a grant from Tom Sawyer Software under NIST Advanced Technology Program.

as to what constitutes a *natural cluster*. There is some intuitive understanding of what constitutes a cluster but there is no universally accepted formal definition of a *natural cluster*. In case the nodes and edges of the graph have some semantic information associated with them (in the form of labels), such information can be used for the purpose of clustering (or *grouping*) the nodes. An example of such information could be the IP addresses associated with the nodes in a telecommunication network. In case the graph has no such information, then the structural properties of the graph have to be utilized for the purpose of generating the clusters. Several candidates for the structures to be used as clusters have been proposed in the literature. These include *biconnected components*, *paths and triangles*, *circles of cliques* and others [1, 2, 5, 6].

2 Clustering using Distance-k Cliques

In spite of the differences of opinion as to what constitutes a cluster, one idea is universally accepted: the nodes belonging to a cluster must have a *strong relationship* between them in comparison with the nodes outside the cluster. Now, we are confronted with another question and that is how to measure the *strength* of a relationship ? In this paper we measure the strength of a relationship between two nodes in a graph in terms of the *distance* between the nodes. The distance between two nodes in a graph is the length of the shortest *path length* between the two nodes. If $dist(u, v)$ gives the distance between the nodes u and v, and if $dist(p, q) > dist(r, s)$, then we say that the strength of the relationship between p and q is *weaker* than the strength of the relationship between r and s. In other words, the strength of a relationship between two nodes is *inversely* proportional to the distance between them.

A subset V' of the node set V of a graph $G = (V, E)$ is defined to be a *Distance-k Clique* if every pair of nodes in V' is connected in G by a path of length at most k. The standard graph theoretic term, *clique*, is a special case of a distance-k clique with $k = 1$. It may be noted that a distance-k clique of a graph $G = (V, E)$ is a subgraph of G with *diameter* k.

In our clustering problem, we would like to partition the node set V of the graph $G = (V, E)$ into fewest number of distance-k cliques. The decision problem version of our clustering problem can be stated as follows:

Partition into Distance-k Cliques Problem
INSTANCE: Given a graph G=(V, E) and positive integers j and k, $1 \leq j, k \leq |V|$.
QUESTION: Can the vertices of G be partitioned into $i \leq j$ disjoint sets $V_1, V_2, \ldots, V_i$ such that, for $1 \leq m \leq i$,the subgraph induced by V_m is a distance-k clique ?

It is not difficult to realize that the *Partition into Distance-k Cliques* problem is NP-Complete because if $k = 1$, it reduces to the *Partition into Cliques* problem, which is known to be NP-Complete [4].

As Partition into Distance-k Clique problem turns out to be an NP-Complete problem, we look for heuristic solutions to the problem that will produce a *good* solution rather than an *optimal* solution. In the following paragraph we describe a class of such algorithms. All these algorithms are variations of one main algorithm which is referred to as the algorithm A1.

2.1 Description of the Algorithms

The clusters are are referred to as $C_1, C_2, \ldots, C_p$ for some integer p. Every node v of the graph $G = (V, E)$ belongs to one cluster $C_i, 1 \leq i \leq p$. This information is stored in an array called $C[1, .., n]$. If for a node $v \in V$, $C[v] = j$, it implies that the node v is an element of the cluster C_j.

A node v of the graph $G = (V, E)$ is known as an *bridge node* if $v \in C_i, (v, u) \in E$ and $u \notin C_i$, where C_i represent the cluster i.

The *neighbors* of a node $v \in V$ is defined as the nodes that are adjacent to v in the graph $G = (V, E))$, i.e., $N(v) = \{u | (v, u) \in E\}$.

Every bridge node $v \in V$ will have a *cluster-list* associated with it. The cluster-list associated with node v, is the set of all the clusters where the neighbors of v, not in $C[v]$, belong, i.e., $CL(v) = \{C_i | N(v) \cap C_i \neq \emptyset \; and \; i \neq C[v]\}$.

The *estimated diameters* of the clusters is an upper bound of the diameter of the clusters. They are stored in an array called $EstDia$. $EstDia(C_i)$ gives the estimated diameter of the cluster C_i.

We first describe the Algorithm A1 and then its variations, algorithms A2 through A5. In all the algorithms, the user specifies the value of k for the algorithms to create distance-k cliques.

Algorithm A1

As a first step, A1 constructs an initial clustering of the nodes. The algorithm InitialCluster1 is used for this purpose. InitialCluster1 chooses high degree nodes of the graph and forms clusters around those nodes. The set of initial clusters called $ClusSet$ and it contains the clusters $C_1, C_2, \ldots, C_p$ for some integer p. This procedure also computes the estimated diameter of each cluster in $ClusSet$ and stores in array $EstDia$. The InitialCluster1 procedure also identifies the bridge nodes at this stage of clustering and for each node $v \in V$ determines the cluster in which this node belongs. This information is stored in array C array,

A list of clusters is associated with each bridge node. The bridge node that has the largest number of nodes in its cluster list is used to (tentatively) form a larger cluster (*ComClus*), comprising of the cluster in which the bridge node belongs and all the clusters in its cluster list. If the estimated diameter of the new cluster is less than or equal to k, then the new cluster is made permanent and the cluster list of all the affected bridge nodes are updated. In case the diameter of the new cluster is greater than k, then such a cluster cannot be formed. In this case, one cluster, (*RemClus*), is removed from the tentative cluster (*ComClus*) so as to reduce the overall diameter and the cluster list of all the affected bridge nodes are updated. The whole process is repeated till the cluster list associated with each bridge node ($CL(i)$) becomes empty.

Algorithm InitialCluster1 (V, E)

begin
$V' = V; P = 0;$
while$(V' \neq \emptyset)$
begin
$P = P + 1;$
m = the highest degree node in V';
$Cluster(P) = \{m\};$
$V' = V' - \{m\}$
$\forall (m,n) \in E$ **do**
begin
if $n \in V'$ **then**;
begin
$Cluster(P) = Cluster(P) \cup \{n\}$
$V' = V' - \{n\};$
end
end
end
end

Algorithm 1: Generation of Clusters using Distance-k Cliques

begin
(S1:) Form IntialClusters; (Suppose at this stage there are
q bridge nodes, $u_1, u_2, \ldots, u_q$.)
for $i = 1$ to q ;
begin
$CL(i) = \{C_j | N(u_i) \in C_j \; and \; j \neq C[u_i]\};$
$|CL(i)| = \sum_{C_j \in CL(i)} |C_j|;$
($|C_j|$ gives the number of nodes in the cluster C_j)
end
while(true);
begin
If $\forall i, 1 \leq i \leq q$, $CL(i) = \emptyset$ then **EXIT**;
(S2:)Find u_m such that the $|CL(m)|$ is the *largest* among all $CL(i), 1 \leq i \leq q$;
$Clist = CL(m) \cup \{C(u_m)\};$
Create a new cluster $ComClus$ and set $ComClus = \{v | v \in C_i \; and \; C_i \in Clist\}$;
Find the clusters with the largest and second largest estimated diameter
in the $Clist$ and call them LC_1 and LC_2;
(S3:)$EstDia(ComClus) = EstDia(LC_1) + EstDia(LC_2) + d$, where $d = 1$
if $LC_1 = C(u_m)$ or $LC_2 = C(u_m)$, otherwise $d = 2$;
if $EstDia(ComClus) \leq K$ **then**
begin
$ClusSet = ClusSet \cup \{ComClus\} - \{C_i | C_i \in Clist\};$
$\forall v \in ComClus$ set $C[v] = ComClus$;
for $i = 1$ to q;
begin
$\forall C_x \in Clist$ if $C_x \in CL(i)$ then remove C_x from $CL(i)$;

If at least one C_x is removed from $CL(i)$ and
if $EstDia(ComClus) < K$ then $CL(i) = CL(i) \cup ComClus$;
end
if $EstDia(ComClus) = K$ **then**
$\forall i$ such that $u_i \in ComClus$ set $CL(i) = \emptyset$;
end
else
begin
if ($LC_1 \neq C(u_m)$ then set $RemClus = LC_1$
else set $RemClus = LC_2$;
if $LC_1 = C(u_m)$ or $LC_2 = C(u_m)$ then
begin
$\forall u_i \in ComClus$ set $CL(i) = CL(i) - RemClus$;
$\forall u_i \in RemClus$ set $CL(i) = CL(i) - ComClus$;
end
else
$CL(m) = CL(m) - RemClus$;
end
end
end

Algorithm A2

This is essentially same as algorithm A1, except that the step marked S2 in A1, should be replaced by the following step:

(S2:)Find u_m such that the $|CL(m)|$ is the *smallest* among all $CL(i), 1 \leq i \leq q$;

This step allows smaller clusters to grow in parallel, as opposed to the algorithm A1 where one cluster grows to its maximum size before another cluster gets an opportunity for growth in size.

Algorithm A3

This again is essentially same as algorithm A2, except that in step marked S1, the procedure InitialCluster2 should be used for initial clustering instead of the procedure InitialCluster1. In InitialCluster2, each node forms its own cluster and that is taken as the starting point of the algorithm A3. It may be noted that in this case every node $v \in V$ is a bridge node if $degree(v) > 0$.

Algorithm InitialCluster2 $(V, E), (V = \{v_1, \ldots, v_n\})$

begin
for $i = 1$ to n **do**
$Cluster(i) = \{v_i\}$;
end

Algorithm A4

This again is essentially same as algorithm A2, except that in step marked S3, instead of estimated diameter of *ComClus*, the *exact* diameter of *ComClus* is computed. This modification gives a higher quality solution at an increased computational cost.

Algorithm A5

This again is essentially same as algorithm A3, except as in algorithm A4, instead of estimated diameter of *ComClus*, the *exact* diameter of *ComClus* is computed. Just like in A4, this modification gives a higher quality solution at an increased computational cost.

2.2 Analysis of Algorithms

We analyze the algorithm A1. The procedure InitialCluster1 can be carried out in $O(n^2)$ time where n is the number of nodes in the graph. The procedure InitialCluster2 can be carried out in $O(n)$ time. The first for-loop can be carried out in $O(n^2)$ time. The while-loop continues till the cluster list associated with each bridge node becomes empty. A bridge node may become a non-bridge node during execution of the algorithm. However, a non-bridge node will never become a bridge node. The number of bridge nodes can be at most n. Inside the while-loop all steps upto and including step marked S3 can be carried out in $O(n)$. The If-statement following S3 is either true or false. If it is true, the statements inside the begin-end block can be carried out in $O(n\Delta)$ time where Δ is the maximum node degree of the graph. If it is false, the statements within the begin-end block can be carried out in $O(n)$. If the begin-end block associated with the true part is executed once, it reduces the number of bridge nodes by at least one. If the begin-end block associated with the false part is executed Δ times, it is guaranteed that the number of bridge nodes will reduce by at least one. Suppose during a particular run, the true part is executed y times and the false part is executed z times. As a result of y time execution of the true part, at least y bridge nodes will cease to be bridge nodes. The remaining $q - y$ bridge nodes must become non-bridge nodes before the while-loop exits. This might require at most $(q - y)\Delta$ executions of the false part of the If-statement. Therefore the total computation in the while-loop will be $O(n) + O(yn\Delta + n(q - y)\Delta)$ = $O(nq\Delta)$. Since the nuber of bridge nodes q and the maximum node degree Δ can be as high as $O(n)$, the worst case complexity of the algorithm will be $O(n^3)$.

3 Experimental Results and Discussions

We tested our implementation of the distance-k clique based clustering algorithm with a wide range of graphs - small, medium and large. We extensively used the Bell Laboratory graph library for testing purpose. This library has a large collection of graphs of wide range of variation in terms of number of nodes, edges and node degrees. The results of the output of our clustering algorithm on Bell Lab graphs 1487 and 1572 with different values of k is presented in table 1. Visualization of the clustering algorithm results is presented in figures 1 through 6. One way to measure the quality of our heuristic algorithm is to find the proximity of the solution produced by it to the optimal solution. We use the following strategy to compute the quality of our solutions: Suppose that the

Table 1. Performance comparison between five algorithms.

Number of clusters generated by different algorithms								
Graph	No. of Nodes	Diameter	k	A1	A2	A3	A4	A5
Bell Lab 1487	48	9	4	14	14	13	11	5
			5	14	11	7	5	3
			6	13	10	8	4	2
Bell Lab 1572	65	13	5	14	12	15	9	7
			7	12	9	12	7	2
			9	14	7	9	4	2

number of clusters generated by heuristic algorithm $A_i, 1 \leq i \leq 5$ for a given graph $G = (V, E))$ and a specified value of k, is $NC_{i,k}$. Suppose for the same graph and same specified k, the optimal number of clusters is OC_k. If we know OC_k, we can easily find its proximity to $NC_{i,k}$. However, OC_k in general is not known. However, we can make the following observation about OC_k: $OC_k \geq diameter(G)/k$. We evaluated the performance of our heuristic algorithms using this criteria. The performance of A_5 is much better than the performance of A_1 at the expense of higher computational time. The performance of the algorithms A1, A2 and A3 is poor in comparison with A4 and A5 because the error in estimated diameter keeps accumulating, resulting in a larger number of clusters. It may be noted that as the objective of the distance-k clustering algorithm is to cluster the nodes that have strong relationship between them, unlike many other clustering algorithms, it does not necessarily minimize the number of intercluster edges.

References

1. F. J. Brandenburg, M. Himsolt and K. Skodinis, "Graph Clustering: Circles of Cliques," *Proceedings of Graph Drawing Symposium, GD'97* Rome, September 1997. Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1353, pp. 158-168, 1998.
2. J.S. Deogun, D. Kratsch and G. Steiner, "An approximation algorithm for clustering graphs with dominating diametral paths," *Information Processing Letters*, 61, pp. 121-127, 1997.
3. P. Eades, "Multilevel Visualization of Clustered Graphs," *Proceedings of Graph Drawing'96*, Berkeley, California, September,1996.
4. M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman Publishers, San Francisco, 1978.
5. D.W. Matula and L.L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *Journal of the Association of Computing Machinery* 30 pp. 417-427, 1983.
6. T. Roxborough and A. Sen, "Graph clustering using multiway ratio cut," *Proceedings of Graph Drawing Symposium, GD'97* Rome, September 1997. Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1353, pp. 291-296, 1998.