Hierarchical Multi-level Approach to graph clustering

by:
Neda Shahidi neda@cs.utexas.edu
Cesar mantilla, cesar.mantilla@mail.utexas.edu

Advisor:
Dr. Inderjit Dhillon

Introduction

Data sets can be presented as set of nodes connected by weighted edges according to some similarity measure. The goal of graph clustering is to identify elements from a data set that are similar or well connected, and assign them to a particular cluster, such that the original graph is divided in several clusters. The problem of graph clustering has been well studied in the five last decades (Ref 1), as a result, the literature is rich in several approaches with different measures of quality of the clusters and computational efficiency, such as multi-level, k-means and spectral methods. Image segmentation problems can also be formulated as a graph partitioning problem and are efficiently solved with spectral methods although their computational cost for large graphs can make them prohibitive. We will focus our attention to the multilevel approach presented by Dhillon et. al. (Ref 3,4 and 5) that used the equivalence between spectral and kernel k-means methods to develop a fast multi-level algorithm. The aim of this project is to implement their algorithm in a hierarchical manner until reaching a base level where the eigen-decomposition is computationally feasible. We applied this approach to segment an image to a basic level and compared it with the regular partitioning from Graclus.

Hierarchical Clustering

Hierarchical schemes are a multilevel decomposition of the original graph in a tree structure producing nested clusters at different levels either from top to bottom or vice-versa. In the top to bottom, the top level is a single level that comprises all the nodes in the graph; this super-cluster is split into two nested clusters in a lower level according certain criterion for partitioning. Each nested cluster is again sud-divided into two in another lower until reaching the desired number of nodes at the lowest level. The bottom to top way, which is called agglomeration, starts merging two similar nodes, decreasing the number of clusters by one after each merging until obtaining a single cluster or the desired number of clusters. These tree structures that are represented by dendrograms showing different hierarchical levels of division. Depending on the application, in some situations, the natural structure of the graph suggests its partition in different levels following a hierarchy. For example, a company that is divided in departments formed by teams of workers, each person is a node connected to others according their functions and responsibilities. On the other hand, if the final number of clusters is known beforehand, a flat clustering would be more adequate (Ref 1) where all clusters are at the same single level.

Spectral Methods for Clustering

Here we recall some definitions of matrices used to represent a graph. The adjacency matrix (A) of a graph is a (NxN) matrix that contains the weights of the edges between nodes, if the nodes are not connected then the weight is zero, where N is the total number of nodes in the graph. The degree matrix (D) is a diagonal matrix whose elements are the sum of the weights of the edges connected to the ith node. The Laplacian matrix of a graph is D-A; finally, the normalized Laplacian, which was proposed by Chung (Ref 8) is D-1/2 L D-1/2. Spectral graph theory relies on the eigen-decomposition of the latter matrix (normalized L), using the eigenvectors of L for finding the clusters like a principal component analysis. Spectral methods work well when the normalized cut or ratio cut are the criterion for the separation of the clusters. However, the main drawback of spectral methods is that they are computationally expensive for large graphs due to the number of operations involved for finding the eigen values of the Laplacian, in some cases unmanageable. However, Dhillon, et. al. generated a program called Graclus that uses a weighted kernel k-means objective equivalent to spectral clustering, rendering the process feasible for large matrices.

Graph Based Image Segmentation

Image segmentation is one of the applications of graph clustering. It would be easy for human criticizer to evaluate the result of segmentation by just looking at the segmented image. Nevertheless, the main reason that we are evaluating our algorithm using image segmentation is that the problem is naturally hierarchical in which main objects in the image form higher level clusters and the small details are in the lower levels. Several methods have been introduced for extracting affinity graph from an image. We have examined two of them:
Method1(Ref 9): In this method each pixel which is a node in the affinity graph can have a common edge only with pixels in its 8-neighborhood. The weight of each edge has inverse relationship with the difference between intensities of two nodes.
Method2(Ref 10): In this method each pixel which is a node in the affinity graph can have a common edge with any pixel in the neighborhood with a pre-specified radius. The weight of the edge in the graph has inverse relationship with the number and strength of crossing edges in the image.

Methodology of this Project

The approach of this project is as same as multi-level clustering except that hierarchical clustering has been used at the base level. We expect to get two advantages from this method:

1. Inherit advantages of hierarchical clustering (no need to know the number of clusters a priori and building clustering tree in naturally hierarchical problems).

2. Decreasing time complexity of refinement stage.

The hierarchical clustering at the base level can be done in either divisive or agglomerative ways. Since the coarsening stage is before this stage, therefore there is no need to modify it. The tricky part of the algorithm is the refinement stage in which we have to decide about how to exchange data between clusters. The following is the proposed algorithm:

Starting with the highest level of hierarchy, exchange data between each two clusters in the same level only if they have the same parent.

Comparing this algorithm with the refinement stage in original multi-level algorithm, we shall see that the number of times each cluster may be involved in an exchange data step is less. But, how the hierarchical structure enables us to do this? The key idea to answer this question is to think about hierarchy as nested clusters. Clearly exchanging data between two clusters makes sense only if they are adjacent. Since we have already captures topology of clusters in the hierarchy, the adjacency of clusters is known. Thus, we can skip some refinement steps for non-adjacent clusters. We will see soon that the time complexity of refinement step is O(n) (in which n is the number of clusters) for our approach while it is $O(n^n)$ for original multi-level approach.

Complexity Analysis of this Method

Time Complexity of Original Multi-level Approach:

Since the refinement stage has to be done between each two clusters, the total number of refinement steps for each coarsened level is c(n,2) which is $O(n^2)$.

Time Complexity of our method:

There are at most 2^i clusters in each level and each one can have one or zero sibling. Therefore the number of refinement steps in ith level is at most $2^i/2$. From this, total number of refinement steps would be

sum($2^i/2$) for i=1,2,...log n = 1+2+4+8+....+2log n -1 < 1+2+4+8+...+n/2 =2*n/2-1=O(n)

This predicts that we can expect even faster refinement stage for our algorithm in spite of that the refinement stage has already been optimized very well in the original multi-level algorithm.

Progress and the Current Results

First Phase

In the first phase of the project, Graclus has been called recursively to build the hierarchy divisively. The objective is to sequentially divide a large graph G0 into two sub-graphs at lower level (k=2), and then each of these graphs be divided into two and so on, until reaching the base level. The resulting two graphs from a division not necessarily have the same number of nodes, thus the process can finalize at different levels in each branch of the tree. Because of the mismatch between the format of input and output of Graclus, a Matlab script has been written to process intermediate data. (see pseudo code in Appendix).

Benchmark Selection

Several matrices from the University of Florida Sparse Matrix Collection have been selected for evaluation of this method. As it is clear from the spy graphs, there are some levels of hierarchy in all of them (Figure

1).

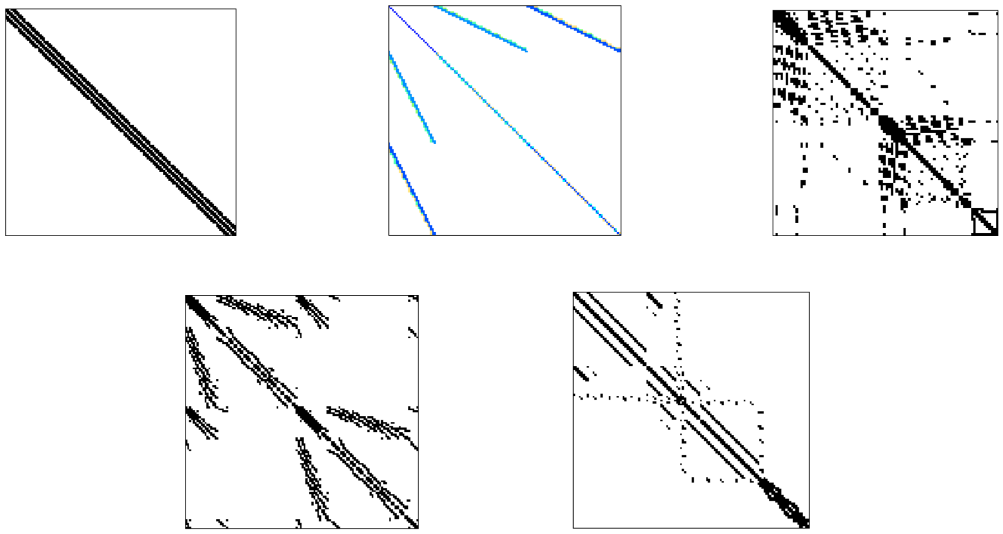


Figure 1 From left to right: Spy of pcryst02, add32, can268, **bottom:** can 836, dwt592.

In addition, some small size images have been selected to be segmented using hierarchical-multi-level method (Figure 2). For converting the image to affinity matrix we tried both methods introduced before. The first method doesn't make the goods results (Figure 3) mainly because it captures very local information from the image. On the other hand, in the second method the locality of information is tunable using neighborhood radius value. In addition, since it works based on the edges between image segments not the intensity of pixels, it works well for images like the two rose pictures in figure 2 in which there are not significant differences between colors of pixels in different segments.

Figure 2 Left to right: Part of a rose with 2500 pixels (we call it Rose50) and Rose with 10000 pixels (we call it Rose100).

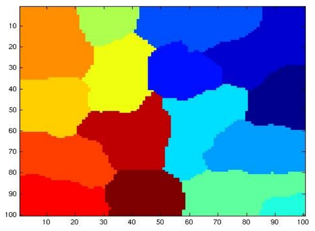


Figure 3 Result of clustering based on affinity matrix from method 1 for Rose100. The number of clusters is 16.

Visualization and Examples

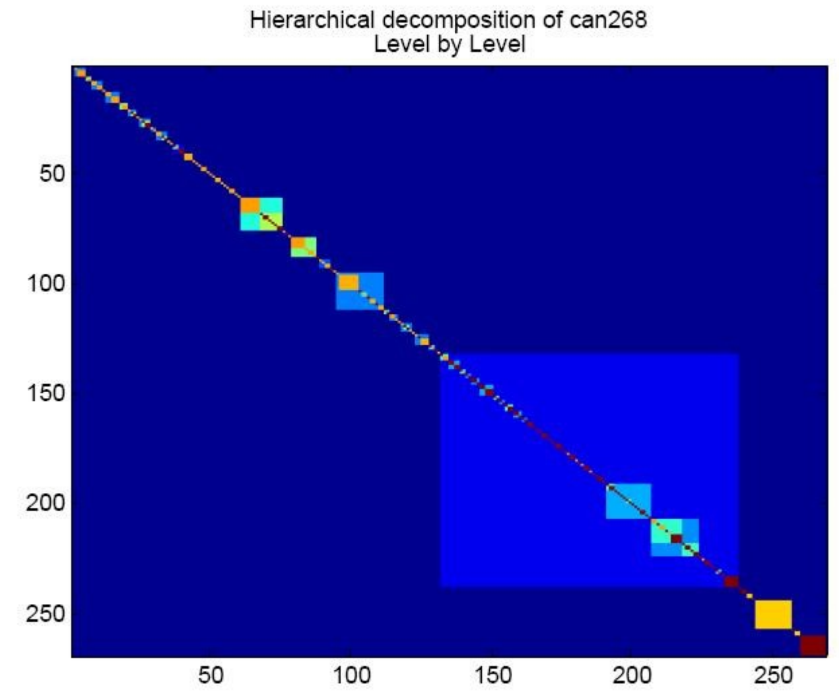
Visualization of hierarchical structures is now in the state of the art of information visualization (Ref 11). Many of methods have been focused on visualization of hierarchy as tree like structures (2D or 3D trees, fractals, etc.). The method we are using here has this advantage that it shows the result of algorithm on the same way that the affinity matrix on graph has been depicted i.e. spy graphs. let's assume that we make a complete graph inside each cluster (just for the visualization purpose) in each level of hierarchy. This generates some full squares in the spy graph of the affinity matrix. We have done this for each hierarchy level and have plotted all of them in the same picture (Fig 4a). The lower level clusters should cover the higher levels, because, otherwise, the big clusters on the higher levels will cover the lower level smaller ones. The resultant visualization of the segmentation of can_268 matrix has been depicted in Fig 4b.

Hierarchical Clustering Experiments and Comparison

These preliminary experiments were intended to compare the goodness of the proposed hierarchical scheme, if the results were better, it worth to modify the Graclus code to have the hierarchical option. The matrices used for our examples were downloaded from the University of Florida Sparse Matrix Collection. Figure 4a shows the spy plot of can_268 that was decomposed until reaching a level of 20 nodes per cluster. Figure 4b shows the progress of the process level by level, initially large clusters are formed (large squares), but as the clusters are smaller at each level smaller squares are formed within the upper level. A common measure of the quality of the clustering is the normalized cuts. We compared in table I the regular clustering with Graclus versus our hierarchical approach for several graphs, showing lower cut ratio for regular Graclus.

Graph File	Min Nodes	Clusters	Cut Ratio Graclus	Cut Ratio Hierarchical
pcryst02	100	204	100.63	106.259
add32	50	163	20.26	17.016
can_268	20	18	7.4981	7.294711
can_838	20	62	28.9745	28.1694
dwt_592	20	42	14.8406	14.503739

Table I. Comparison of the cut ratio by Graclus and hierarchical partitioning



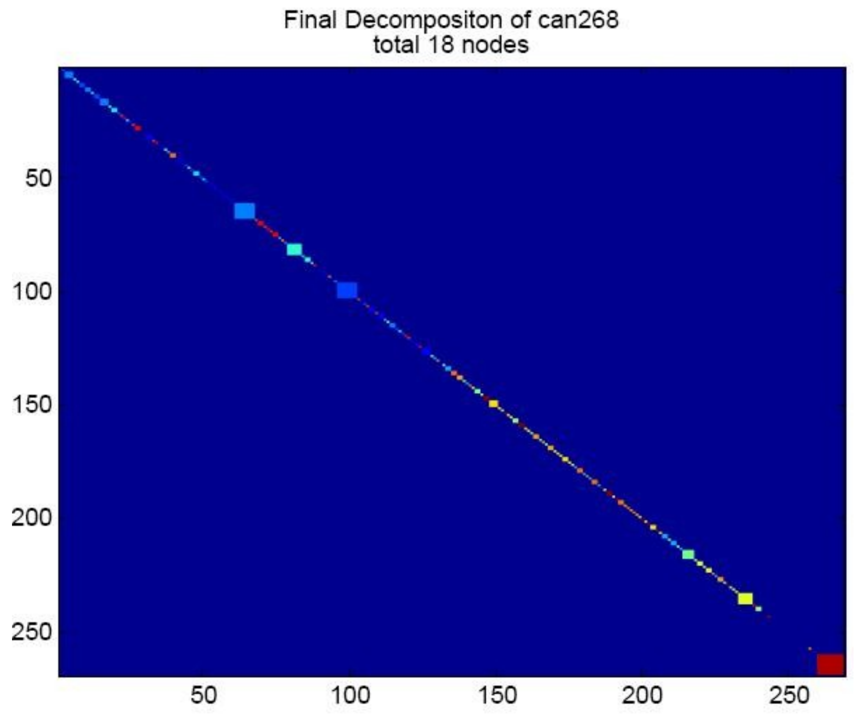


Figure 4. From top to bottom: partitioning progress plot and final clustering for matrix can_268

A simple graph illustrative example

The following example serves to illustrate the differences between both methods. In the first hierarchical level (Figure 5b), tough the number of cuts is minimum (3), one of the natural clusters is broken. It would be preferable to break by the dashed line as a logic choice, but still the partition in red minimizes the number of cuts. In the second level, (Figure 5c) the partition in the upper super cluster creates a 6-nodes super cluster that in the third and last level is broken, resulting in a two-nodes cluster in the upper corner. The total number of cuts in the hierarchical approach was 9, whereas with the regular Graclus was 8, and the clusters are better balanced. The smallest cluster has 3 nodes as oppose to the two-node cluster resulting from the hierarchical approach. The base clustering method of Graclus tends to generate balanced clusters, what in this case helped to reduce the cut ratio. In summary, the regular Graclus outperformed our hierarchical approach because it does optimize the partitioning considering the graph as a whole structure, while independent partitioning loses the connection between sibling graphs. In general we can infer that since divisive hierarchical clustering takes divide and conquer approach, it inherits the advantages like simplicity of solving sub-problems. However, the chance of falling into local optima will increase as well.

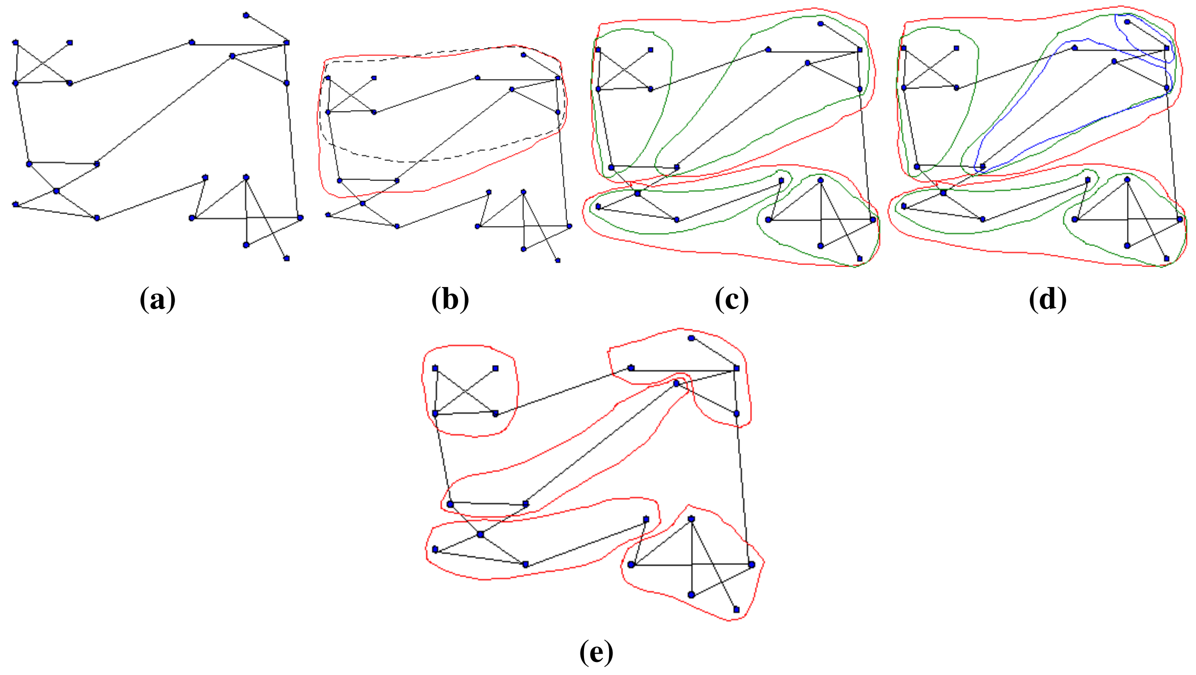


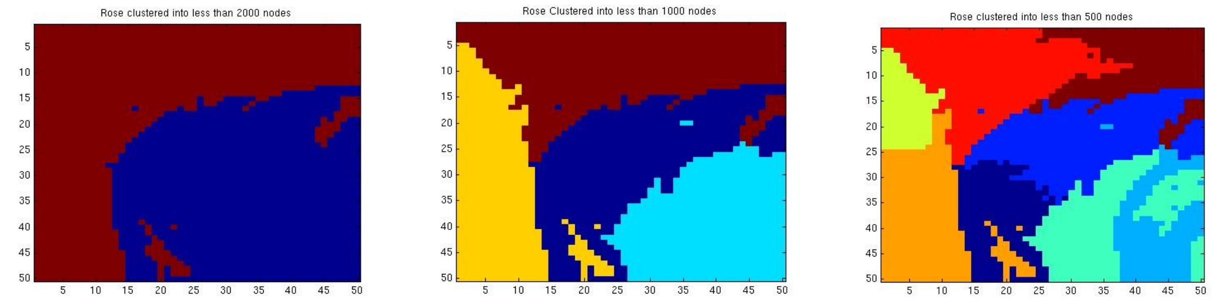
Figure 5 (a) Original graph, **(b)** first level, **(c)** second level, **(d)** third level and **(e)** final partitioning with hierarchical and regular Graclus respectively.

Image Example

We experimented partitioning the small square shown of the Rose50 at different hierarchical levels as presented in the series of figure 6. When it was divided into clusters with less than 300 nodes, it ended at different levels. We also partitioned the Rose100 as shown in figure 7 and have compared it to simple graclus result, but the computational time is excessive to make several cases.

Where we are going from here

As we mentioned, the result of hierarchical clustering method can be improved if we modify Graclus to cluster at the coarsest level because then the refinement step will be run on all of the clusters at the same time. This may give a second chance to the solution to overcome some local optima. Furthermore, in the implemented method, hierarchical clustering has to be done in divisive manner while agglomerative methods are more popular. Therefore in the next step, we will try agglomerative clustering. We also will explore the relation between the Laplacian matrices of the graphs at the lowest level and the original graph to see how good they can represent the original graph.



2 nodes 4 nodes 8 nodes

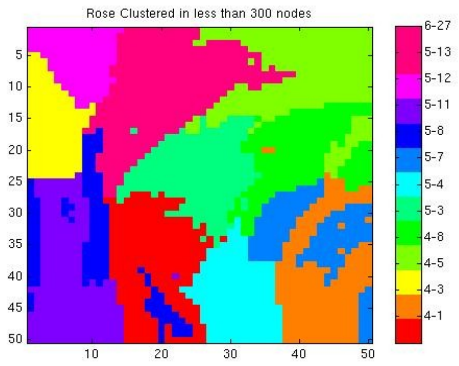


Figure 6. Clustering of Rose50 in hierarchical levels according to the number of nodes at the base level

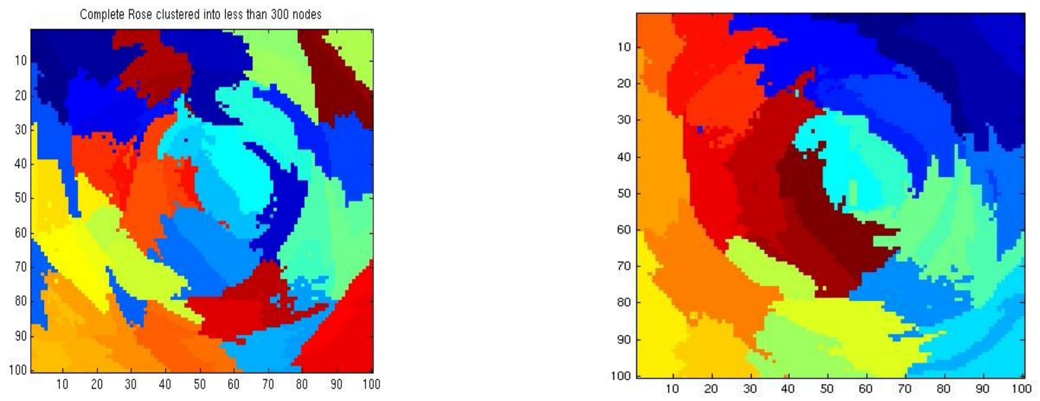


Figure 7. **left to right:** Complete Rose100 partitioned in clusters with less than 300 nodes, partitioning using simple gruclus to 28 clusters.

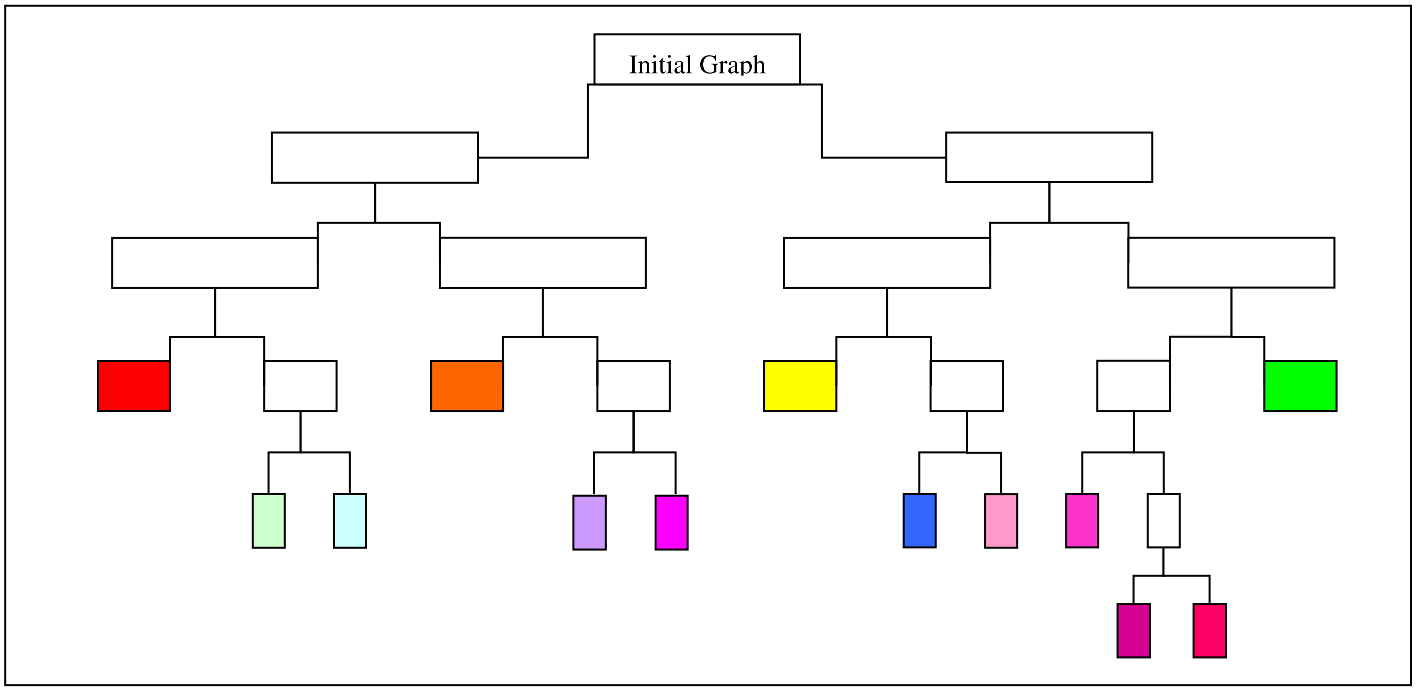


Figure 8. Clustering the Rose50 ended in clusters with less than 300 nodes at different levels as indicated by the color scale.

References:

1.Schaeffer, A. E. " Survey: Graph clustering", Computer Science Review I (2007) 27-64.

2.Flake, G. W., Tarjan, R. E. and Tsioutsiouliklis, K. "Graph Clustering and Minimum Cut Trees", Internet Mathematics, Vol I, No. 4 385-408.

3.Dhillon, I. S., Guan,Y. and Kulis, B. "Weighted Graph Cuts without Eigenvectors: A Multilevel Approach", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 29, No. 11, Nov 2007.

4.Dhillon, I. S., Guan,Y. and Kulis, B. "A Fast Kernel based Multilevel Algorithm for Graph Clustering", *Proceedings of* The 11th ACM SIGKDD, Chicago, IL, August 21 - 24, 2005.

5.Dhillon, I. S., Guan,Y. and Kulis, B. "Kernel k-means, Spectral Clustering and Normalized Cuts", *Proceedings of* The 10th ACM SIGKDD, Seattle, WA, August 22-25, 2004.

6.G. Karpys and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs", SIAM, J. Scientific computing, Vol. 20. No. 1 359-392, 1999.

7.B. Hendrickson and R. Leland, " A Multilevel Algorithm for Partitioning Graphs" Technical Report SAND93-1301, Sandia national Laboratories, 1993.

8.Chung, F.R.K. Spectral Graph Theory, American Mathematical Society, Providence, RI. USA, 1997

9.Felzenszwalb P.F., and Huttenlocher D.P., "Efficient Graph-Based Image Segmentation", International Journal of Computer Vision, Volume 59, Number 2, September 2004.

10. Yu S.X., Gross R. and Shi J., "Concurrent Object Recognition and Segmentation by Graph Partitioning", Neural Information Processing Systems, Vancouver, 2002.

11. Robertson, G. G., Hierarchy Visualization: From Research to Practice, Lecture Notes In Computer Science, NUMB 3843, pages 533-534, 2006.

Appendix:

Pseudocode for hierarchical Partitioning

```
Read Initial Graph file (G1_1);
Level=0;
While (#Nodes of all Graphs>Base Level)
    Increase Level;
    For i=1 to 2^(Level)
        If (Nodes of Graph(Level, i) >Base Level)
            Partition (Graph(Level, i)) ➔ Graph (Level+1,2*i-1) and Graph(Level+1,2*i);
        Else
            Label  Graph(Level,i) as done;
        End
    End
End
```