Topological Decomposition and Heuristics for High Speed Clustering of Complex Networks

Faraz Zaidi*, Guy Melançon**

*Karachi Institute of Economics and Technology (KIET)
Korangi Creek, Karachi, 75190, Pakistan
faraz@pafkiet.edu.pk
**CNRS UMR 5800 LaBRI & INRIA Bordeaux - Sud Ouest
351, cours de la Libération, 33405 Talence cedex, FRANCE
guy.melancon@labri.fr

Abstract. With the exponential growth in the size of data and networks, development of new and fast techniques to analyze and explore these networks is becoming a necessity. Moreover the emergence of scale free and small world properties in real world networks has stimulated lots of activity in the field of network analysis and data mining. Clustering remains a fundamental technique to explore and organize these networks. A challenging problem is to find a clustering algorithm that works well in terms of clustering quality and is efficient in terms of time complexity.

In this paper, we propose a fast clustering algorithm which combines some heuristics with a Topological Decomposition to obtain a clustering. The algorithm which we call Topological Decomposition and Heuristics for Clustering (TDHC) is highly efficient in terms of asymptotic time complexity as compared to other existing algorithms in the literature. We also introduce a number of Heuristics to complement the clustering algorithm which increases the speed of the clustering process maintaining the high quality of clustering. We show the effectiveness of the proposed clustering method on different real world data sets and compare its results with well known clustering algorithms.

1 Introduction

Most real world systems take the form of networks where a set of nodes and edges might be used to represent these networks. Examples include social networks, metabolic networks, food web, transport networks (Newman (2003)). Clustering remains an important technique towards the better exploration and organization of these networks. In terms of networks representing real world data, a cluster can be defined as a group of nodes which are similar or connected in some predefined sense and dissimilar to nodes belonging to the other clusters (Schaeffer (2007)). Detection of clusters has a wide range of applications in various fields. For example, in social networks, clustering could lead us towards a better comprehension of the interactions taking place between people, or for biological networks, a useful application of clustering is in the identification of biomarkers in a protein-protein interaction network.

Different measures have been studied to classify these networks. Two such classifications have gained lots of interest when networks exhibit small world (Watts and Strogatz (1998)) and scale free (Barabási and Albert (1999)) features. These features make the networks complex and the clustering problem challenging. Some examples of networks that are both scale free and small world at the same time are the network of Author (Newman (2001)) and Movie Actor network (Watts and Strogatz (1998)).

Another important issue that needs to be addressed while developing clustering algorithms for these networks is the Time Complexity as with the increasing size of these networks, it becomes almost impractical to use slow clustering algorithms. Algorithms exist in the literature addressing the clustering problem for large size complex networks but a trade off exists between Clustering Accuracy and the Time Complexity. Thus it is evident that faster algorithms are required to achieve high speed clustering as well as high accuracy to handle large networks.

The motivation of this work comes from the fact that node degree distribution of real world networks is not random, rather different nodes have varying degrees. Specially with the presence of scale free behavior, lots of nodes tend to have few connections whereas a few nodes dominate the network connectivity with a large number of connections. These networks form a single connected component but a careful analysis suggests that nodes having high degree, play an important role in keeping the entire network connected. Using a Topological Decomposition of networks based on degree, we propose a new clustering algorithm which is quite efficient in terms of time complexity and performs as well as existing clustering algorithms in terms of the quality of clustering produced. We also introduce some high speed heuristics that help to reduce the size of the network in liner time in terms of number of nodes.

Throughout this article, we use the term network to refer to an undirected and unweighted simple graph represented by G. We represent the number of nodes by n and the number of edges by m. Rest of the paper is organized as follows: Section 2 discusses a number of clustering algorithms present in the literature. In section 3, we explain the details of the topological decomposition. We then introduced the TDHC algorithm in section 4. In section 5, we present real world data sets used for experimentation. We compare the results of the TDHC algorithm with existing algorithms in section 6, finally concluding in section 7.

2 Related Work

Many different approaches have been proposed to discover clusters in complex networks. For example, Girvan and Newman (Girvan and Newman (2002)) used edge betweenness to produce a divisive hierarchical clustering algorithm. The basic idea is to identify intra cluster edges as compared to inter cluster edges. Edges lying between clusters will have a higher betweenness centrality as compared to edges within a cluster. The clustering algorithm removes edges with high betweenness centrality to identify clusters and recalculates the betweenness centrality. The algorithm performs well in the detection of clusters but suffers from high time complexity. The worst case time complexity is given by $O(m^2n)$. Although practically the algorithm runs faster than its worst case but still has a high time complexity due to the calculation of betweenness centrality in each iteration since in each iteration, the total number of nodes are divided by a certain factor before recalculating the betweenness centrality.

Wu et al.(Wu et al. (2004)) introduce a multilevel mesh structure to cluster large networks. The clustering algorithm uses Betweenness centrality and node degree to identify a set of representative nodes. All the other nodes are assigned the nearest representative nodes to obtain clusters. The agglomerative process is repeated to obtain a hierarchical clustering which they call multilevel mesh. At each level, the user chooses a branching factor which determines the number of clusters for that level. This number might not represent the actual number of clusters in the dataset as they are determined by the user without the use of any heuristic or statistical measure. The overall complexity of the algorithm is given by $O(m^2n)$.

Boccaletti et al. (Boccaletti et al. (2007)) propose a clustering method based on the cluster de-synchronization properties of phase oscillators. Starting from a fully synchronized state of the network, a dynamical change in the weights of the interactions that retain information on the original betweenness distribution, yields a progressive hierarchical clustering that fully detects the dense communities. Since the initial calculation of betweenness takes $O(n^2)$, the algorithm scales quadratically as the number of nodes increase.

Newman (Newman (2004)) presents a faster agglomerative hierarchical clustering algorithm which is based on a quality function called modularity Q. The algorithm repeatedly joins communities together in pairs, choosing at each step, the join that results in the greatest increase in Q. The time complexity for the algorithm is given by $O((m + n)n)$ which scales quadratically in terms of number of nodes in the graph.

An important class of clustering algorithms called Spectral Clustering algorithms have attracted considerable interest (Spielman and Teng (1996)). The biggest advantage of these algorithms is that they are able to detect clusters without a specific form as compared to classical algorithms such as k-means. Moreover they are well suited for large size networks as well. But these algorithms are suited only of data sets where the similarity graphs are sparse (Luxburg (2007)). For graphs having scale free properties, where a few nodes are connected to lots of nodes, results in non-sparse similarity graphs. An example of the type of graphs we have with small world and scale free properties is shown in Figure 1(a). The graph is layed out using a Force Directed algorithm (Hachul and Jünger (2005)). These algorithms are well known to put densely connected nodes, closer to each other and sparsely connected nodes distant to each other. From the figure, it is quite clear that the algorithm fails to do this with small world and scale free graphs due to the presence of high degree nodes. Another draw back of spectral clustering algorithms is that the results are highly dependent on the choice of initial parameters and different parameters can result in big changes in the clustering (Luxburg (2007)). Selecting correct parameters require the user to be well aware about the data and the clusters to be generated which can be problematic.

Wu and Huberman (Wu and Huberman (2004)) model a network as an electrical circuit and the clustering algorithm is based on the notion of voltage drops across networks. The idea is that each edge is considered to be a resistor between two nodes. By solving Kirchhoff's equations (Alexander and Sadiku (2008)) the voltage value can be obtained for each node. Using this voltage value, the community of the node can be determined. Although the total running time of the algorithm is $O(m + n)$ but the algorithm has to be repeated a certain number of times to achieve a certain precision. Another algorithm that performs well in terms of execution time is based on a heuristic method that optimizes modularity (Blondel et al. (2008)). The algorithm does not use normalized modularity which is considered to be a flaw (Fortunato and Barthélemy (2007)).

Efficient algorithms to cluster networks with only small world properties have been proposed like (Auber et al. (2003); van Ham and van Wijk (2004)). These systems perform well if the topology of the network follows small world properties but fail to perform in the presence of scale free properties. This is due to the fact that in a scale free network, a few nodes dominate the entire networks connections and makes it difficult to identify the clusters.

3 Topological Decomposition of Graphs

In this section, we describe a method introduced earlier by authors (Zaidi and Melançon (2010)) to detect the presence of densely connected nodes in a network in relatively quick time. The method is based on a decomposition technique which exploits the fact that nodes having high degree are responsible for keeping large size networks as a single connected component.

To decompose the network into several components, Max$_d$-Degree Induced Subgraphs (Max$_d$-DIS) are constructed where Max$_d$-DIS is an induced subgraph constructed by considering only the nodes having degree at most d in graph G. Mathematically for a graph $G(V, E)$ where V is a set of nodes and E is a set of edges, the Max$_d$-DIS is defined as $G'(V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$ and $\forall u \in V', Deg_G(u) \leq d$ where d can have values from 0 to the maximum node degree possible for a network. We construct Max$_d$-DIS for $d = \{0, 1, \cdots, MaxDeg\}$ to obtain a set of graphs $(G_0, G_1, \cdots, G_{MaxDeg})$. Construction of a Max$_d$-DIS can be achieved in $O(n)$ time and if it is repeated for all possible values of node degree, the process can be performed in $O(n * MaxDeg)$ time where $MaxDeg$ is the maximum possible degree of a node in graph G. Consider the example of the Author network shown in Fig. 1. The entire network is shown in Fig. 1(a), where as Fig. 1(b) shows a small portion being focused where the encircled nodes represent densely connected nodes or more precisely cliques. Fig. 1(c) and (d) show portions of the Max$_3$-DIS and Max$_5$-DIS drawn using a force directed algorithm (Hachul and Jünger (2005)). In these two figures, it is quite easy to visually detect the cliques or the densely connected nodes.

The inspiration of our clustering algorithm comes from this visualization. Fig. 1(c) and (d) clearly show that nodes get disconnected in the absence of high degree nodes and these disconnected components can be easily identified as subgraphs. We argue that from these subgraphs, the problem of finding clusters can be simplified as a problem of counting number of edges and number of nodes in a connected component. Calculating a connected component is a problem that can be solved in $O(n + m)$ time. Similarly the counting of nodes and edges also runs in linear time. Keeping in mind that n and m can be very small depending upon the value of d chosen, this step runs quite faster then its worst case as there are only a limited number of nodes and edges in Max$_d$-DIS as compared to the entire graph G.

4 Proposed Clustering Method: TDHC

From the topological decomposition, the idea of building a clustering algorithm is quite intuitive. Since in these subgraphs, we can identify the set of nodes that are densely connected to each other in quick time, they can be grouped to form clusters. So a hierarchical clustering algorithm can be built which calculates the Max$_d$-DIS for varying values of d and groups the densely connected nodes. The notion of how to define density is addressed later in this section. The number of iterations do not depend on n or m of G but on a factor of maximum degree a node can have in G. Along with the detection of densely connected nodes through

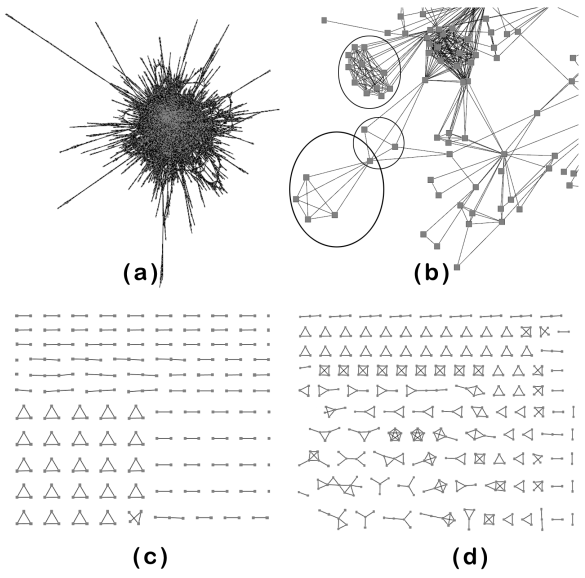

FIG. 1 – *Co-Authorship Network (a) Entire Network (b) Focus on a Small Portion (c) Part of Max_3-DIS (d) Part of Max_5-DIS*

Max_d-DIS, we also introduce several heuristics that optimize the performance of the clustering algorithm. Note that the heuristics only improve the convergence speed of the algorithm to a single cluster, and the basic algorithm can be executed without using these heuristics. All these steps are highly efficient in terms of time complexity and are discussed below.

Sink using K-Sink Operation: We define the K-Sink operation as follows: The nodes having degree 1 in a network suggests that they are only connected to a single node. We merge the 1-degree nodes into their neighbors creating a new node for each such merger. The 1-degree nodes merged into the neighbors are called the Sinkers. The nodes in which the 1-degree nodes get merged are called the Sinkholes. This operation is justified because a 1-degree node cannot be clustered with any other node as it is simply connected to only one node. We call this operation, a 1-Sink operation and it is illustrated in Fig. 2(a) where node 2 is the sinker and node 1 is the sinkhole. If two nodes have degree 1 and are connected to each other, this means that they are disconnected from the rest of the network and in this case either of the node can be chosen to be the sinker and the other as the sinkhole.

Similarly we define a 2-Sink operation, consider two nodes, say node 2 and node 3 both having a degree 2 (Fig. 2(a)). They are connected to each other, and to another node say node 1, with a higher degree, nodes 2 and 3 can be sinked into node 1 as they are only connected to either each other or node 1. This operation is illustrated in Fig. 2(a) and we call this 2-Sink operation as Type A. Just as in the case of 1-Sink, if we find a set of nodes each having degree exactly equal to 2 and connected to each other, this means that they are not connected to the rest of the graph, in this case any node can be chosen to be the sinkhole and the other two nodes to be the sinker. Another type of 2-Sink operation, Type B, is when a node of degree 2,

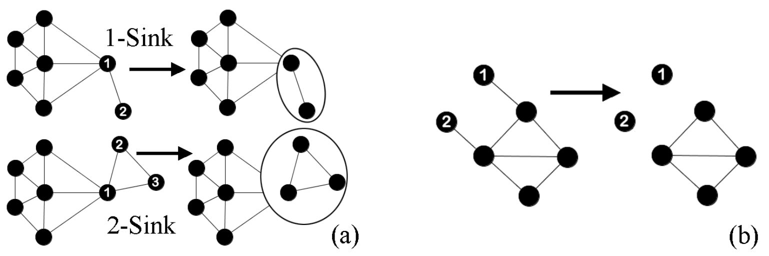

FIG. 2 – *(a)K-Sink operation illustrated with 1-Sink and 2-Sink operations. (b) Tightening Operation where Nodes 1 and 2 get disconnected leaving the other nodes densely connected.*

is connected to two other nodes of degree more than 2. Irrespective of whether these two high degree nodes are connected to each other or not, the two degree node can only be clustered with either one of these two nodes. What we do is simply put the two degree node with the neighbor having a highest degree and create an edge between this cluster and the other neighbor.

For the implementation of the algorithm, we only use 1-Sink and 2-Sink operations although the idea can be generalized to sink nodes up to some constant K. Both 1-Sink and 2-Sink operations can be performed in time $O(n)$. But a generalized implementation to incorporate K-Sink operation will no longer remain linear and since our goal is to keep the time complexity bounded by a linear function or as close as possible to a linear function we avoid using a generalized K-Sink operation. The order in which these K-Sink Operations were performed is important, where first we perform a 1-Sink operation, followed by a Type A and Type B 2-Sink Operations. Then the Type A 2-Sink operation is repeated finally followed by a 1-Sink Operation. Rememeber, in most real world networks, the node degree distribution is not random, but is exponential. And there are lots of nodes with low node degree and only a few nodes with high node degree. The K-Sink operation is desinged to cluster these nodes quickly, the basic idea being that if a node is connected to a single node, it should be clustered immediately with this node. We repeat that this heuristic is quite logical and does not effect the quality of clustering, unless singleton clusters are allowed to be generated, which might not be interesting for a domain expert to analyze.

Maximum Degree Induced Subgraph: The next step in the algorithm is to create a Max$_d$-DIS with a small value of d. Due to this small value, the network might break into several components disconnected to each other as shown in Fig. 1(c) and (d).

Tightening: Disconnect Loosely Connected Nodes: After obtaining the Max$_d$-DIS, we perform an operation that we call Tightening. We look at the nodes having degree 1 in this subgraph and we simply remove the edges connecting degree 1 nodes from the induced subgraph as shown in Fig. 2(b). This process helps us to make the connected components found in the subgraph more denser. Moreover, since it is not certain whether the 1 degree nodes in the Max$_d$-DIS do actually belong to the cluster of the node with which it is connected, this step ensures that nodes are only assigned to clusters that they belong to. The step can be easily performed in time $O(n)$ where n can have small values as compared to the entire graph G.

Calculation of Connected Components: Once we have the Max$_d$-DIS, we calculate all the connected components in the subgraph. We use a breadth first search algorithm (BFS) starting from a node and iterating through its neighbors to find the connected component it belongs to. Once we have identified nodes connected to the start node, we restart the BFS from a node that has not yet been visited. The algorithm runs in $O(n + m)$ time.

Grouping Densely Connected Components: The final step is to group the connected components that are densely connected to each other. We explain how to evaluate if the component is dense enough later in this section. Once we have found the densely connected components in the subgraph, we cluster these nodes in graph G. We replace this cluster of nodes with a single node in G. Multiple edges connecting this new cluster node to other nodes are removed to make sure that the graph remains simple. We only consider components of size greater than 2 nodes to be clustered together.

Clustering Algorithm: Now that we have explained all the necessary steps, the Topological Decomposition and Heuristics for Clustering (TDHC) is presented as algorithm 1. The algorithm starts by calculating a MaxD$_2$-DIS in order to search for triangles representing three nodes and connected to each other. For nodes having degree 1, they get sinked in the 1-Sink step and thus we do not need to run the algorithm for MaxD$_1$-DIS. Note that in the algorithm, when a step is performed on G, the size of G in terms of number of nodes is decreased as nodes within G are grouped together to form clusters.

Algorithm 1 TDHC Algorithm

```
$Input\ G(V, E)$
$d \leftarrow 2$
$increment \leftarrow 1$
while $Number\_of\_Nodes(G) > 1$ do
    K-Sink($G$)
    $G'$ = Create_Max$_d$−DIS($G$)
    Tightening($G'$)
    Calculate_Connected_Component($G'$)
    Group_Densely_Connected_Component($G'$)
    $d \leftarrow d + increment$
end while
```

All the processing steps have a linear time complexity as shown in previous sections. The number of iterations required to converge towards a solution no longer depends on the number of nodes nor the edges but on the maximum degree a node can have. Moreover, as in the given algorithm, we have chosen an increment of 1 at every iteration, in this case, the algorithm executes at most d times. The choice of the value for the variable $increment$ depends on the user, which can be increased depending on how the results vary as a function of this value. A high increment value means less number of iterations, but risks in less dense components found. Currently we have kept the increment value to 1, but we intend to experiment with this parameter in the future to study the variation in the quality of clusters produced.

The average case time complexity of the entire algorithm can be expressed as $O(d{*}(m{+}n))$ where d is the maximum degree of a node in graph G. An important observation about the clustering algorithm is that it uses both the Divisive as well as Agglomerative approaches to cluster graphs. The divisive part comes from the fact that we build degree induced subgraphs and the agglomerative part is represented when we cluster nodes during K-Sink operation and grouping densely connected components.

Flattening the Clusters: The hierarchical clustering thus produced can have many clusters with 2 or 3 nodes due to the K-Sink operation explained earlier. We simply parse recursively through different clusters to remove these small size clusters and merge them into bigger size clusters. To produce a partitional (flat) clustering, using the same algorithm, all we need to do is replace the condition in the algorithm where we want to converge to a single node by the number of clusters we want to obtain in the network. Once we get to this number, we can flatten the hierarchy to obtain a partitional clustering. We have used this same approach to compare the results of the TDHC algorithm with the other clustering algorithms.

Density Function: There are several definitions of how to calculate the density of a graph (Melançon (2006)). For simplicity we use the node to edge ratio (n/m) to refer to the density of the graph. Melançon (2006)) argues that the density of a graph varies as a function of application domain giving real world examples. For the proposed clustering algorithm, we use a density function to determine how well a set of nodes is connected to each other. Based on the arguments and examples provided in (Melançon (2006)), we argue that we cannot have a generic density value set as a threshold to decide whether a set of nodes is connected enough or not. Moreover, the question of whether a set of nodes are connected well enough to be clustered, depends not only on the density of the entire graph but on the underlying structure of the network as well.

To resolve this problem, we propose a floating density function i.e. we propose a set of functions starting from high density values to progressively less dense functions. The idea is to try to find highly dense communities first, for all possible values of the Max$_d$-DIS, and then replace the density function with a less denser function. We start by looking for the maximum number of edges possible for a set of nodes and eventually end up looking for the minimum number of edges possible for a set of nodes to be connected. We cluster a set of nodes if the number of edges m is:

$m = n(n-1)/2$ $\qquad$ $m \geq n(n-1) * 0.9/2$
$m \geq n(n-1) * 0.6/2$ $\qquad$ $m \geq n(n-1) * 0.4/2$
$m \geq (1.5 * n) - 0.5$ $\qquad$ $m \geq n$

The set of equations represent a gradual decrease in the node-edge density required for a group of nodes to be considered as dense enough to be clustered together. Although using the floating equation idea can effect the number of iterations required to cluster the entire data set, but it assures us that the clusters found would be dense. This is the only control parameter that is required by the proposed algorithm and varies from one dataset to the other. The overall complexity of the algorithm remains the same as the number of equations ranges from a constant value of 2 to 6.

5 Experimentation

Data Sets: The first data set is the Co-Authorship network of scientists working on network theory and experiments(Newman (2006)). The second data set is a network mapping data which consists of paths from a test host towards other networks on the Internet containing routing and reachability information (`www.opte.org`). Since the Divisive Clustering algorithm has a high time complexity, we only consider a subset of the actual data with 1049 nodes and

1319 edges. The third data set is a Protein interactions network used by (Gavin (2002)). The data is available from the website(`http://dip.doe-mbi.ucla.edu/dip`) and contains 1246 nodes and 3142 edges. Disconnected nodes (80 nodes) were removed from the data. The choice of these data sets is based on the criteria that all these networks belong to different classification of networks as described in the literature (Newman (2003)). The *author* network represents a *social network* of collaboration, the *internet* network represents a *technological network* and the *protein* network represents a *biological network*. All these networks have an exponential (not necessarily following power-law) degree distribution. The Clustering Coefficient of the author network is 0.74 and the average path length is 6.04, that of internet network is 0.005 and 6.42, and finally for the protein network is 0.23 and 4.89 respectively.

Clustering Algorithms: To cluster these data sets, we use two known clustering algorithms, the Bisecting K-Means algorithm (Steinbach et al. (2000)) and the Divisive Clustering algorithm based on Edge Centrality (Girvan and Newman (2002)). The choice of these algorithms is based on the criteria that these algorithms do not try to optimize or influence the clustering algorithm based on the density or some other cluster quality metric as compared to other algorithms present in the literature such as (Newman (2004)). Moreover they are known to perform well for a number of real world data sets (Girvan and Newman (2002)). We also use the Strength Clustering algorithm proposed by (Auber et al. (2003)). The algorithm has been shown to perform well for the identification of densely connected components as clusters.

Cluster Evaluation Metrics: To evaluate the quality of clustering produced, we use the following metrics. Modularity(Q) (Newman and Girvan (2004)) (Q metric) is a metric that measures the fraction of the edges in the network that connect within-community edges minus the expected value of the same quantity in a network with the same community divisions but random connections between the vertices. If the number of within-community edges is no better than random, we will get $Q = 0$. Values approaching $Q = 1$, which is the maximum, indicate strong community structure. The second metric used by Auber *et al.* (Auber et al. (2003)) is called the MQ metric. It comprises of two factors where the first term contributes to the positive weight represented by the mean value of edge density inside each cluster. The second term contributes as a negative weight and represents the mean value of edge density between the clusters. Finally the Relative Density (RD) (Mihail et al. (2002)) of a cluster calculates the ratio of the edge density inside a cluster to the sum of the edge densities inside and outside that cluster. The final RD is the averaged sum of the these individual relative densities for all clusters.

6 Results and Discussion

As shown in the previous sections the algorithm's average case time complexity is $O(d * (m + n))$, but in reality, the algorithm runs much faster than its average case. This is because as the algorithm progresses, the nodes are aggregated into clusters and the size of the network becomes smaller. We compare the results of the TDHC clustering algorithm with (Girvan and Newman (2002); Newman (2004); Auber et al. (2003)) in Table1. From the different values, it is quite clear that the TDHC algorithm performs as well as the other clustering algorithms. Although using the RD metric, its performance is not as good as the other clustering algorithms. These differences highlight the behavior of various cluster evaluation metrics present in the literature. Nevertheless, considering the time complexity of TDHC compared to the

	Author			Internet			Protein		
Algorithm	MQ	Q	RD	MQ	Q	RD	MQ	Q	RD
Div. Clus.	0.53	0.77	0.63	0.32	0.79	0.69	0.31	0.63	0.49
Bis. K-Means	0.42	0.77	0.63	0.41	0.59	0.58	0.41	0.33	0.31
Strength	0.83	0.26	0.23	0.50	0.35	0.55	0.52	0.16	0.29
TDHC	0.55	0.82	0.42	0.42	0.85	0.49	0.38	0.44	0.23

TAB. 1 – *Results of Divisive Clustering based on Edge Distribution (Div. Clus.), Bisecting K-Means (Bis. K-Means) and Strength Clustering algorithms with the TDHC algorithm.*

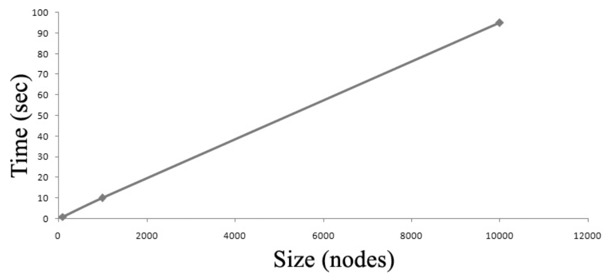

FIG. 3 – *Linear Running time of the TDHC algorithm with increasing graph size.*

other algorithms, empirical results of TDHC show that the algorithm performs well on different data sets. We do not claim that our algorithm produces better quality results for different types of networks and cluster evaluation techniques but we show that our algorithm performs as well as other algorithms. The major contribution of the algorithm is the low asymptotic time complexity which enables us to run the algorithm for large size networks.

Figure 3 shows the execution time of TDHC algorithm for graphs of increasing size in terms of number of nodes. The graphs were generated using artificial network generation model for small world and scale free graphs using the model of Klemm and Eguiluz (2002).

Analyzing the algorithm, we try to exploit two important characteristics of networks, the degree distribution and the clustering coefficient. The Topological decomposition uses the fact that real world networks do not have a uniform degree distribution, thus the decomposition helps to break the network into several components. And on the other hand, the networks having high clustering coefficient represent the presence of densely connected nodes in the network, which can be grouped together to form clusters. The idea of floating density function works well for networks that do not have high clustering coefficient (See Internet Network) as we try to group nodes which are less densely connected. The results show that the algorithm performs well for different types of networks.

7 Conclusion and Future Research Directions

In this paper we have used Heuristics and a technique based on the Topological Decomposition of the network to develop a high speed clustering algorithm. The low asymptotic time complexity of the algorithm opens new horizons to the domain of network analysis and clustering. As shown by the results, the proposed algorithm performs as well as other existing algorithms in terms of accuracy but largely out performs them in terms of time complexity.

From this study, there are many questions that need to be further explored in detail and presents new and challenging research opportunities. For example the K-Sink operation as an important utility to reduce the complexity of scale free networks and clustering them based on this operation only. The Max$_d$-DIS as an important decomposition of small world networks for clustering. We intend to perform extensive study using the presented topological decomposition and expect to find new and interesting results.

References

Alexander, C. and M. Sadiku (2008). *Fundamentals of Electric Circuits*. McGraw-Hill.

Auber, D., Y. Chiricota, F. Jourdan, and G. Melancon (2003). Multiscale visualization of small world networks. In *INFOVIS '03: Proceedings of the IEEE Symposium on Information Visualization*, pp. 75–81.

Barabási, A. L. and R. Albert (1999). Emergence of scaling in random networks. *Science 286*(5439), 509–512.

Blondel, V. D., J.-L. Guillaume, R. Lambiotte, and E. Lefebvre (2008). Fast unfolding of communities in large networks. *J. Stat. Mech. 2008*(10), P10008+.

Boccaletti, S., M. Ivanchenko, V. Latora, A. Pluchino, and A. Rapisarda (2007). Detection of complex networks modularity by dynamical clustering. *Physical Review E 75*.

Fortunato, S. and M. Barthélemy (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences 104*(1), 36–41.

Gavin (2002). Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature 415*(6868), 141–147.

Girvan, M. and M. E. J. Newman (2002). Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA 99*, 8271–8276.

Hachul, S. and M. Jünger (2005). Drawing large graphs with a potential-field-based multilevel algorithm. *Graph Drawing*, 285–295.

Klemm, K. and V. M. Eguiluz (2002). Growing scale-free networks with small world behavior. *Physical Review E 65*, 057102.

Luxburg, U. (2007). Tutorial on spectral clustering. *Statistics and Computing 17*(4), 395–416.

Melançon, G. (2006). Just how dense are dense graphs in the real world?: a methodological note. In *BELIV '06: Proc. of the 2006 AVI workshop on BEyond time and errors*, pp. 1–7.

Mihail, M., C. Gkantsidis, A. Saberi, and E. Zegura (2002). On the semantics of internet topologies, gitcc0207. Technical report, College of Comp., Georgia Institute of Tech.,USA.

Newman, M. E. (2001). Scientific collaboration networks. i. network construction and fundamental results. *Phys Rev E Stat Nonlin Soft Matter Phys 64*(1 Pt 2).

Newman, M. E. and M. Girvan (2004). Finding and evaluating community structure in networks. *Phys Rev E Stat Nonlin Soft Matter Phys 69*(2 Pt 2).

Newman, M. E. J. (2003). Structure and function of complex networks. *SIAM Review 45*, 167.

Newman, M. E. J. (2004). Fast algorithm for detecting community structure in networks. *Physical Review E 69*, 066133.

Newman, M. E. J. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) 74*(3).

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review 1*(1), 27–64.

Spielman, D. A. and S.-H. Teng (1996). Spectral partitioning works: Planar graphs and finite element meshes. In *In IEEE Symposium on Foundations of Computer Science*, pp. 96–105.

Steinbach, M., G. Karypis, and V. Kumar (2000). A comparison of document clustering techniques. Technical report, Dept. of Computer Science and Engineering, Univ. of Minnesota.

van Ham, F. and J. van Wijk (2004). Interactive visualization of small world graphs. In *INFOVIS 2004. IEEE Symposium on Information Visualization*, pp. 199–206.

Watts, D. J. and S. H. Strogatz (1998). Collective dynamics of 'small-world' networks. *Nature 393*, 440–442.

Wu, A. Y., M. Garland, and J. Han (2004). Mining scale-free networks using geodesic clustering. In *KDD '04: Proc. of ACM SIGKDD*, pp. 719–724.

Wu, F. and B. A. Huberman (2004). Finding communities in linear time: A physics approach. *The European Physics Journal B 38*, 331–338. informal publication.

Zaidi, F. and G. Melançon (2010). Identifying the Presence of Communities in Complex Networks Through Topological Decomposition and Component Densities. In *EGC 2010, Extraction et Gestion de Connaissance*, Volume E-19, RNTI. 163-174.

Résumé

Avec l'accroissement exponentiel de la taille des données et des réseaux, il devient nécessaire de développer des techniques nouvelles et rapides d'analyse et d'exploration de ces réseaux. De plus, l'émergence de propriétés du petit monde et graphes sans échelle dans les réseaux du monde réel a grandement stimulé l'activité dans le domaine de l'analyse de réseau et d'exploitation des données. Le regroupement demeure une technique fondamentale pour explorer et organiser ces réseaux. La difficulté consiste à trouver un algorithme de regroupement qui fonctionne bien en termes de qualité de regroupement et qui soit efficace en termes de complexité de temps.

Dans cet article, nous proposons un algorithme de regroupement rapide qui combine certaines heuristiques avec une Décomposition Topologique pour obtenir un regroupement. L'algorithme que nous appelons Décomposition Topologique et Heuristiques pour Regroupement (TDHC) est très efficace en termes de complexité asymptotique de temps comparé aux autres algorithmes existant dans la litérature. Nous introduisons également un nombre d'Heuristiques pour compléter l'algorithme de regroupement qui accroit la vitesse du processus de regroupement en maintenant la haute qualité du regroupement. Nous montrons l'efficacité de la méthode de regroupement proposée sur différentes séries de données du monde réel et nous comparons ses résultats avec des algorithmes de regroupement bien connus.