

An-Najah National University

Faculty of Graduate Studies

Bridging Centrality in Scale-Free Network Using Bridging Node as the Boundary of Clustering

By

Hind Ali Ahmad Eid

Supervisor

Dr . Subhi Ruzieh

*Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Computational Mathematics, Faculty of Graduate Studies, at
An-Najah National University, Nablus, Palestine.*

2010

Bridging Centrality in Scale-Free Network Using Bridging Node as the Boundary of Clustering

By

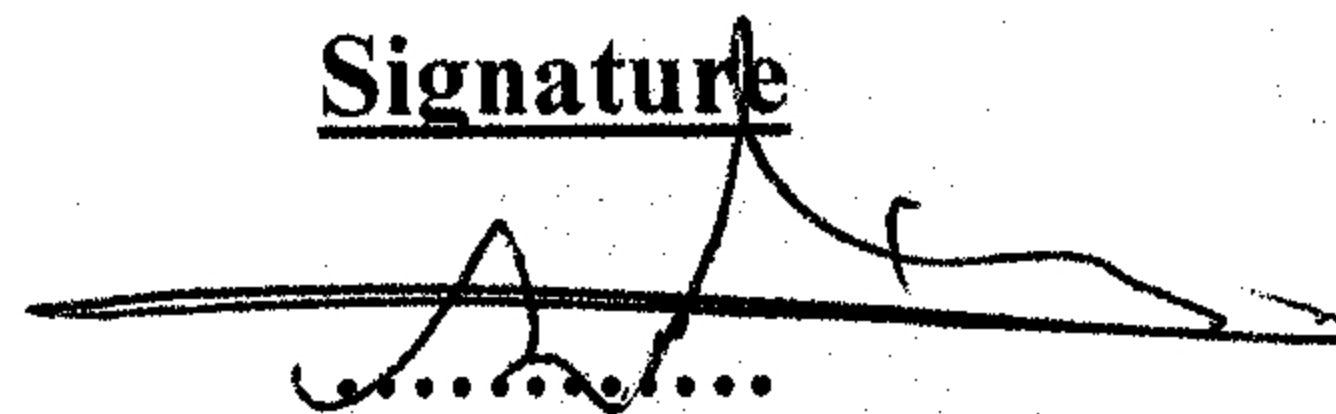
Hind Ali Ahmad Eid

This Thesis was defended successfully on 8/2/2010 and
approved by:

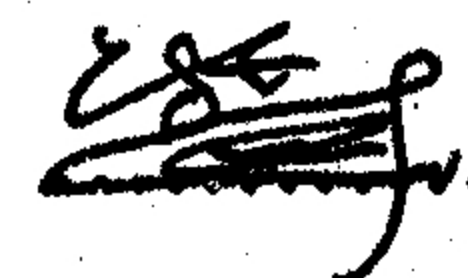
Committee Members

1- Dr. Subhi Ruzieh(Supervisor)

Signature



2- Dr. Saed Mallak (External Examiner)



3- Dr. Mohammad Omran (Internal Examiner)



4- Dr. Naji Qatanani (Internal Examiner)

N. Qatanani

Dedication

I would like to present my thesis to my parents to mother-in-law,
to my husband Haytham Halabi , and to my daughters Haneen
and Azzah .

Acknowledgement

First of all I would like to thank my supervisor Dr. Subhi Ruzieh efforts and important guidance for the completion of this thesis.

الاقرار

انا الموقع ادناه مقدم الرسالة التي تحمل العنوان :

Bridging Centrality in Scale-Free Network Using Bridging Node as the Boundary of Clustering

التجسير المركزي في الشبكات الحرة باستخدام عقدة الجسر كمرجع للتجميع

اقر بأن ما اشتملت عليه هذه الرسالة انما هي نتاج جهدي الخاص. باستثناء ما تمت الاشارة اليه
حيثما ورد، وأن هذه الرسالة ككل، أو أي جزء منها لم يقدم من قبل لنيل اية درجة علمية أو بحث
لدى أية مؤسسة تعليمية أو بحثية أخرى.

Declaration

The work provided in this thesis, unless otherwise references, is the researcher's own work, and has not been submitted elsewhere for any other degree or Qualification.

Student's name:

اسم الطالبة:

Signature:

التوقيع:

Date :

التاريخ:

Table of Content

no	Content	Page
	Dedication	III
	Acknowledgement	IV
	Declaration	V
	Table of Content	VI
	Table of Tables	IX
	Table of Figures	X
	New Accomplish	XII
	Abstract	XIII
Chapter One Introduction		1
1.1	Graph Theory	1
1.2	Main Definitions in Graph Theory	2
1.2.1	Graph	2
1.2.2	Definitions	2
1.3	Scale-Free Network	8
1.3.1	Definition	8

1.3.2	History of Scale-Free Network	9
1.4	Clustering	10
1.5	Bridging Centrality	11
1.6	Highest Bridging Centrality Cut Algorithm	11
Chapter Two Scale-Free Network		12
2.1	Scale-Free Network	12
2.1.1	Definition	12
2.1.2	Comparison Between Complex and Scale-Free Network	13
2.2	Power Law Distribution	19
2.3	Clustering	21
2.3.1	Definition	21
2.3.2	Clustering Method	22
2.3.2.1	Hierarchical Clustering	22
2.3.2.2	Partitioned Clustering	28
Chapter Three Bridging Centrality		37

VIII

3.1	Bridging Centrality	37
3.2	Bridging Coefficient	37
3.3	Betweenness Centrality	39
3.4	Special Cases	49
3.4.1	Complete Graph	49
3.4.2	Complete Bi-partite Graph	50
3.4.3	Star Graph.	52
3.4.4	Cycle Graph	53
3.4.5	Path Graph	56
Chapter Four Clustering Analysis In Unweighted Graph		59
4.1	Unweighted Graph	59
4.2	Highly Connected Subgraph Clustering algorithm	59
4.2.1	Properties of HCS Clustering	62
4.2.2	Modified HCS Algorithm	62
4.3	HCS Analysis	63
4.4	Properties of Cluster	64

4.5	Analysis	65
4.6	Highest Bridging Centrality Cut Algorithm	66
4.7	Properties of HCRC Algorithm	71
4.8	Comparison Between HCS Algorithm and HCRC Algorithm	72
4.9	Conclusion	75
	References	88
	Appendices	91
	Appendix(A) My Matlap Programs	91
	Appendix(B) The Adjacency Matrix For Small Unweighted graph in figure(3.1)	99
	Appendix(C) The Adjacency Matrix For Unweighted Graph in Figure (4.2)	100
	Appendix(D) The Adjacency Matrix For Large Unweighted Graph in Figure (4.13)	101
	الملخص	ب

Table of tables

N o	Table	Pag e
2. 1	The Values of γ_{in} and γ_{out} For Some Scale-Free Networks.	20
3. 1	Values of C_R , BC , C_B For the Graph in Figure (3.1)	48

Table of Figures

no	Figure	Page
1.1	Graph G (5, 7)	2
1.2	Graph G (7,9)	3
1.3	(a) Weighted Graph (b) Unweighted Graph	4
1.4	(a) Directed Graph. (b) Undirected Graph	5
1.5	Complete Graph	5
1.6	Star Graph	6
1.7	Complete Bi-Partite Graph	6
1.8	4-Partite Graph	7
1.9	6-Cycle Graph	7
1.10	Path Graph	7
1.11	Directed Graph G.	8
2.1	Random and Scale-Free Network	12
2.2	(a) Random Network , (b)Scale-Free Network	13
2.3	Difference (1) between Random and Scale-Free Network.	14
2.4	Difference(2) Between Random and Scale-Free Network.	15
2.5	Different (3) Between Random and Scale-Free Network	15

XII

2.6	Map of Italian Cities	24
2.7	Step 1 We Cluster MI and TO in One Item.	25
2.8	In Step 2 We Cluster RM and NA In to One Item.	26
2.9	Step3 We Cluster BA With NA\RM Into One Item.	26
2.10	In Step 4 We Cluster FI With BA\NA\RM.	27
2.11	Steps of Hierarchal Clustering Using Agglomerative and Divisive Method	28
2.12	Graph of the Nodes IN Example(2.5),	31
2.13	Clusters in Iteration Two Example (2.5)	32
2.14	Cluster of Iteration Three Example (2.5)	34
2.15	Cluster of Iteration Four Example(2.5)	35
3.1	Simple Graph	43
3.2	Random Unweighted Graph	47
3.3	5-Cycle Graph	53
3.4	6-Cycle Graph	54
4.1	An Example on HCS Clustering Algorithm	61
4.2	We Can See that the Three Nodes Which Form Cluster 2 in the Previous Example are Taken as Singletons	63

XIII

4.3	Random Unweighted Graph	63
4.4	Scale-Free Network	66
4.5	Applying HCRC Algorithm for Clustering	69
4.6	Small Unweighted Undirected Graph	72
4.7	random unweighted graph	73
4.8	When We Apply HCS Algorithm to Graph in Figure (4.7),	74
4.9	When We Apply HC _R C Algorithm to the Graph in Figure (4.7) We Get Three Clusters Only in Two Step	74
4.10	Large Unweighted Scale-Free Network Example (4.4) step1.	77
4.11	Step 2 in Example(4.4)	78
4.12	Step 3 in Example(4.4)	79
4.13	Step 4 in Example(4.4)	80
4.14	Step 5 in Example(4.4)	81
4.15	Step 6 in Example(4.4)	82
4.16	Step 7 in Example(4.4)	83
4.17	Step 8 in Example(4.4)	84
4.18	Step 9 in Example(4.4)	85

New Accomplishments

The main aim of our thesis was to reach to a new algorithm for clustering depend on the bridging centrality of the nodes, and we reach to this result, and through our research we can reach to many new rules and definitions that we use from to reach to our result.

First: We can obtain a new algorithm to compute the Betweenness centrality of the nodes, to compute the bridging centrality of the nodes.

Second: We Derive special cases of Betweenness centrality, bridging coefficient, end bridging centrality of the nodes for some types of the graph.

Third: We proposed some modification for highly connected subgraph algorithm.

Fourth: We define new properties of cluster, that make number of clusters less and average number of nodes in clusters more. These properties were more suitable exactly when we deal with scale-free network.

Fifth: We reach to new algorithm for clustering which depend on bridging coefficient of the nodes (HCRC Algorithm). In this algorithm we can reach to the same clusters we can reach by using other clustering algorithm, with less running time.

Sixth: Compare our algorithm with other clustering algorithm.

Bridging Centrality in Scale-Free Network

Using Bridging Node as the Boundary of Clustering

By

Hind Ali Ahmad Eid

Supervisor

Dr . Subhi Ruzieh

Abstract

Graph theory is one of the most popular fields in mathematics because of its important applications in solving many problems in the real world and understanding many natural phenomena.

This work focuses mainly on studying the scale-free networks and their properties. Moreover, it deals with the study of clustering methods and developing a new clustering algorithm by using the properties of scale-free networks. Bridging centrality of the graph together with Betweenness centrality and bridging coefficients will also be investigated. Finally we will illustrate how bridging centrality is used in clustering.

This will result in a new algorithm of clustering that is called Highest Bridging Centrality Cut algorithm ($HC_R C$ algorithm). We concluded that the $HC_R C$ algorithm depends on bridging centrality of the nodes.

Chapter One

Introduction

1.1 Graph Theory

In mathematics and computer science, graph theory is the study of graphs: mathematical structures used to model pair wise relations between objects from a certain collection.

In the real world graph theory and its applications can solve many problems such as computer networks, airplane lines, and many other kinds of networks.

By using graph theory applications we can minimize the cost and maximize the benefits. For example we can find the shortest path between any two nodes in any graph. There are many different algorithms which can solve this problem, and when we say shortest path we don't mean the distance only. When we deal with weighted graphs, graphs where edges are assigned weights, we may evaluate a lot of objectives like the minimum cost, the maximum profit, the minimum distance between two locations and many other objectives related to maximizing or minimizing problems. And we can use graph theory applications in organizing steps for solving some problems such as those related to computer applications, by finding the critical path, and many other applications.

1.2 Main Definitions in Graph Theory

1.2.1 Graph

A graph G is an ordered pair $(V(G), E(G))$ where $V(G)$ is a set of nodes (vertices), and $E(G)$ the set of edges, where the number of vertices is the order of the graph and the number of edges is the size of graph G , $G(p,q)$ is a graph with order p and the size of the graph $= q$.

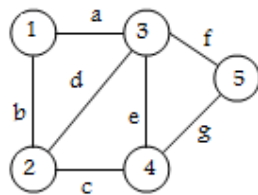


Figure (1.1): Graph $G(5, 7)$

Graph $G(5, 7)$ in Figure(1.1) is a graph of order $= 5$ (it has five Vertices (or nodes)) and of size $= 7$ (it has seven Edges) the set of Vertices $V=\{1,2,3,4,5\}$, and the set of Edges $E=\{a,b,c,d,e,f,g\}$.

1.2.2 Definitions

Let $G(p,q)$ be a graph with order p and size q :

A loop: is an edge whose end vertices are the same vertex.



Multiple edges: are two or more edges joining the same pair of vertices.



A simple graph: is a graph which has neither loops nor multiple edges.

A complex graph: is a graph which has loops or multiple edges or both.

A walk: an alternating sequence of vertices and edges, beginning and ending with a vertex.

A trail: is a walk in which no edge is repeated.

A path: is a walk in which no vertex is repeated.

A closed walk: is a walk of the form (a, b, c, \dots, a) .

A cycle: is a closed path.

Degree of node: is the number of nodes that are adjacent to this node.

Connected Graph: is the graph that has a path between any two nodes.

Disconnected graph: if there is a pair of nodes in graph has no path between them the graph is called disconnected graph.

An isolated vertex: is a vertex with degree 0.

An end vertex: is a vertex with degree 1.

Adjacent: two vertices v, w of a graph G are adjacent if there is an edge between them, and we can say v and w are neighbors, the set of nodes that are adjacent to node v is $N(v)$.

Example 1.1:

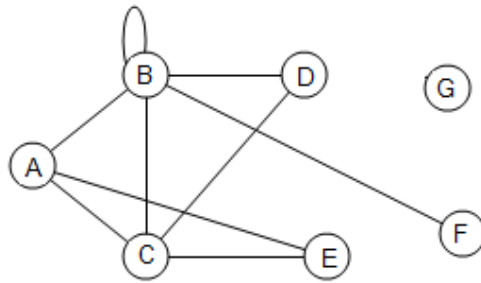


Figure (1.2): Graph $G(7,9)$

Considering the graph in Figure (1.2)

(A, C, D, B, C, E) is a walk. (B, C, D, B, E) is a trail.

(A, E, C, B, D) is a path. (A, B, C, E, A) is a cycle.

F is an end vertex, G is an isolated vertex.

The degree of node C ($\text{DEG}(C) = 4$, $\text{Ne}(C) = \{A, B, D, E\}$).

The graph G in Figure (1.2) is disconnected graph.

There are different kinds of graphs depending on the kinds of application we deal with. We will deal with two kinds of graphs according to the concept adopted.

The first concept of that is related to the weights of the edges in the graph. According to this concept a graph is either weighted or unweighted.

- 1- A weighted graph is when the edges in this graph have values or weights, these values may be lengths, costs, periods or of other different types, these values are called weights. We may for example have a graph whose vertices are the cities in some country and the weights are the distances between the neighboring cities.
- 2- AN unweighted graph: may be thought of as a weighted graphs with all weights being equal to one. An example is a computer network.

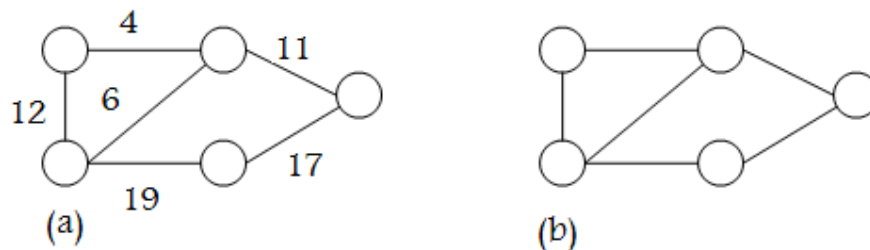


Figure (1.3): (a) Weighted Graph (b) Unweighted Graph

The other concept when dealing with graphs concerns the direction.

Here we look at two types of graphs.

- 1- A directed graph each edge has a specified direction.

2- Undirected graph, we consider that edges to have no restricted direction and the move can be in either one of the two directions.

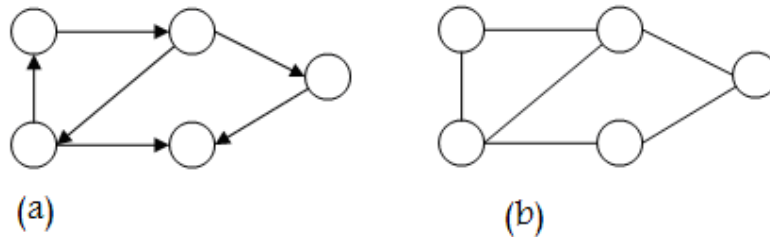


Figure (1.4): (a) Directed Graph. (b) Undirected Graph

There are some special types of graph such as:

- The complete graph $G(p, q)$ every vertex is adjacent to every other vertex. The degree of a vertex in such a case is $p-1$. Such a graph, where all vertices have the same degree is called a regular graph.

Thus the complete graph is a $(p-1)$ -regular graph.

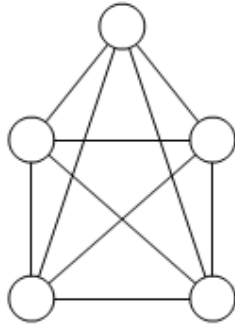


Figure (1.5): Complete Graph.

- Star: the graph take the shape of the star, one node adjacent to all other nodes (degree of this nodes equal $(p-1)$). And the other nodes have degree equal one.

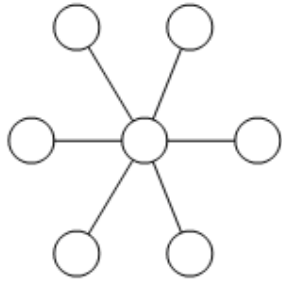


Figure (1.6): Star Graph.

- The complete bipartite graph $K(m, n)$ is a simple graph where the vertex set $V(G)$ is partitioned into two classes of sizes m and n respectively and where every vertex in one class is adjacent to every vertex in the other class. Besides that, any two vertices in the same class are non adjacent. Such a graph clearly has size $q = mn$.

In general a graph is .

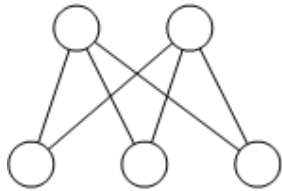


Figure (1.7): Complete Bi-Partite Graph.

- N -partite if is a simple graph where the vertex set $V(G)$ is partitioned into n classes of sizes m_1, m_2, \dots, m_n respectively and where every vertex in any class can be adjacent to any vertex in any other classes. Besides that, any two vertices in the same class are non adjacent. Such a graph clearly has size $q = m_1 m_2 \dots m_n$

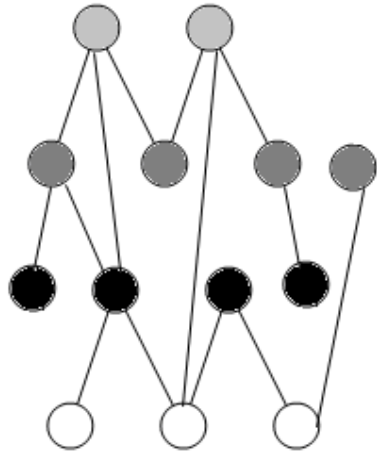


Figure (1.8): 4-Partite Graph.

- The P-cycle C_p is a graph that takes the shape of cycle and every vertex has degree equal to two.

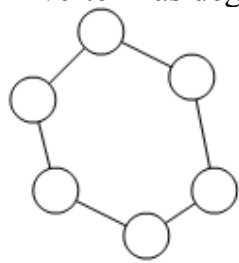


Figure (1.9): 6-Cycle Graph.

- The path is a graph that takes the shape of line. Each vertex has degree equal two except the first vertex and the last vertex each of which has degree equal one

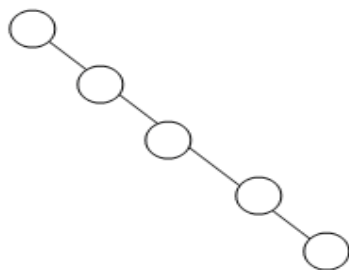


Figure (1.10): Path Graph.

Adjacency Matrix: The adjacency matrix of a graph G is an $n \times n$ matrix $A = a(i,j)$ in which the $a(i,j) = \begin{cases} 1 & \text{nodes } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$

Distance Matrix: The distance matrix of a graph G is an $n \times n$ matrix $D = d(i,j)$ in which the entry $d(i,j) = \begin{cases} l & \text{nodes } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$, l is the distance between nodes i and j .

Example (1.2)

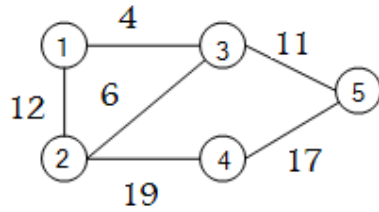


Figure (1.11): Directed Graph G .

Adjacency matrix is $A(G) = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$

The distance matrix is $D(G) = \begin{bmatrix} 0 & 12 & 4 & 0 & 0 \\ 12 & 0 & 6 & 19 & 0 \\ 4 & 6 & 0 & 0 & 11 \\ 0 & 19 & 0 & 0 & 17 \\ 0 & 0 & 11 & 17 & 0 \end{bmatrix}$

1.3 Scale-Free Network

1.3.1 Definition

Scale-free network is a complex connected graph (network) with the property that the number of links k originating from a given node exhibits

a power law distribution $P(k) \sim k^{-\gamma}$ where γ is the degree exponent that varies between 2 and 3.[20]

1.3.2 History of Scale-Free Network

In 1999 the physicist Albert-Laszlo Barabasi and his colleagues at the university of Notre Dame mapped the connectedness of the Web. To their surprise, the Web did not have an even distribution of connectivity (so-called "random connectivity"). Instead, some network nodes had many more connections than the average. In seeking a simple categorical label, Barabási and his collaborators called such highly connected nodes "hubs". In physics, such right-skewed or heavy-tailed distributions often have the form of a power law. I.e., the probability $P(k)$ that a node in the network connects with k other nodes was roughly proportional to $k^{-\gamma}$, and this function gave a roughly good fit to their observed data.

After finding that a few other networks, including some social and biological networks, also had heavy-tailed degree distributions, Barabási and collaborators coined the term "scale-free network" to describe the class of networks that exhibit a power-law degree distribution. Soon after, Amaral et al. showed that most of the real-world networks can be classified into two large categories according to the decay of $P(k)$ for large k . [20]

1.4 Clustering

Clustering is grouping similar data items together.

There are different clustering algorithms. We can divide clustering algorithm into two main categories:

- 1- Hierarchical clustering. This Proceeds successively by either merging small clustering into larger ones or splitting larger clustering.
- 2- Partitioned clustering. This attempts to directly decompose the data set into a set of disjoint clusters.

Each type has different algorithms. A bridging node refers to a node whose removal disconnects the network.

Many scientist study clustering algorithms and they obtain different algorithms, in 1999 Erez Hartuv and Ron Shamer reach to new clustering algorithm called Highly Connected Subgraph clustering algorithm, this algorithm depend on the connectivity of the graph (the connectivity of the graph G (or edges connectivity) is the minimum number of edges whose

11

removal disconnect the graph). After that they modify this algorithm by repeating the algorithm several time until no new cluster appears, because some times the graph has more than one minimum cut and the algorithm may chose the wrong cut.

In our thesis we apply this algorithm on scale-free network but the result was not suitable, exactly in scale-free network there are many end points.

1.5 Bridging Centrality

Bridging centrality is a concept used to identify bridging nodes in scale-free networks. The bridging centrality of node v , $(C_R(v))$, is the product of the betweenness centrality of the node v , $(C_B(v))$, and the bridging coefficient of a node v , $(BC(v))$ (The bridging coefficient of a node determines the extent of how well the node is located between high degree nodes, The betweenness centrality is a measure of the global importance of a node that assesses the proportion of a shortest path between all node pairs that pass through the node of interest).

In 2006 a group of scientist (Woochang Hwangy ,Young-rae

Choy, Aidong Zhangy, and Murali Ramanathan) study the Bridging Centrality in scale-free network, and they reach to a result that bridging nodes lying between highly connected modules in scale-free networks. (Bridging node: is a node that lies between modules in the graph and its removal disconnect the graph, the bridging node have high bridging centrality), in the last of there research the question was if it is possible to reach to a new algorithm of clustering depend on bridging centrality.

1.6 Highest Bridging Centrality Cut Algorithm

After deep studying for clustering method and bridging centrality of the node in scale- free network, and several iteration we can reach to a new clustering algorithm depend on the bridging centrality, we called this algorithm Highest Bridging Centrality (C_R) Cut algorithm IHC_RC algorithm.

Chapter Two

Scale-Free Network

2.1 Scale-Free Network

2.1.1 Definition

Scale-free network is used to give small number of edges high degree so that these nodes are adjacent to 70% or more of the nodes in the network and large number of nodes with small degree)

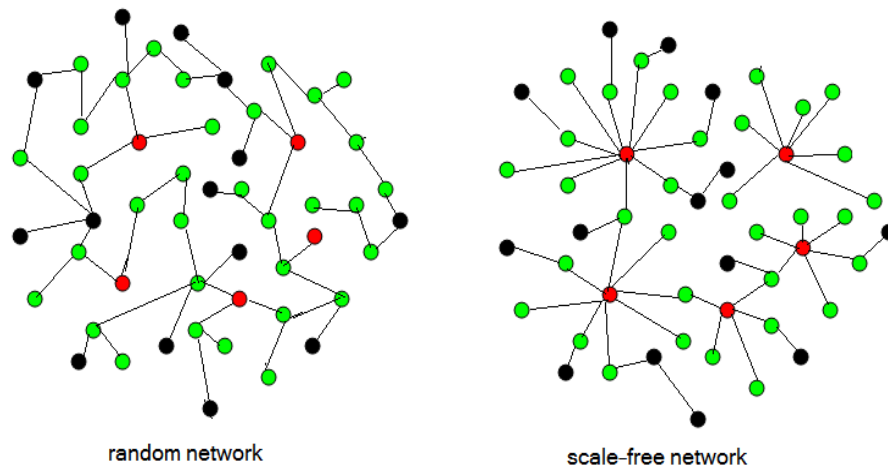


Figure (2.1): Random and Scale-Free Network.

In Figure (2.1) in scale-free network we can see that the red nodes are adjacent to 70% of all other nodes (the green nodes), in random network red points are adjacent to only 10% of all other nodes.

2.1.2 Comparison Between Random and Scale-Free Network

By studying the properties of scale-free network we can find that there are some differences between random and scale-free network:

- 1 Scale-free network is more robust against failure[3]:

This means that if we remove some nodes in random way, scale-free networks are more likely to be connected than random networks.

Example (2.1):

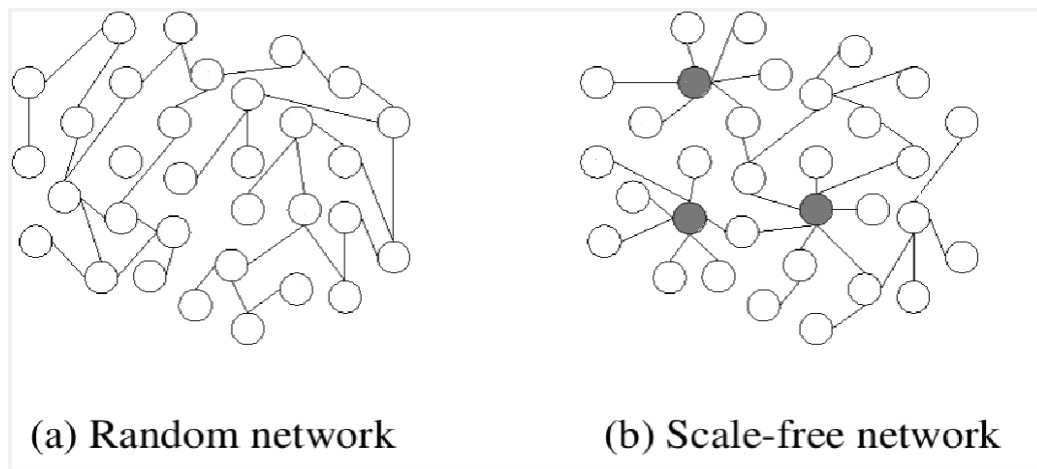


Figure (2.2): (a) Random Network, (b) Scale-Free Network

If we delete some nodes in random way, the scale-free network may be more connected than the random network.

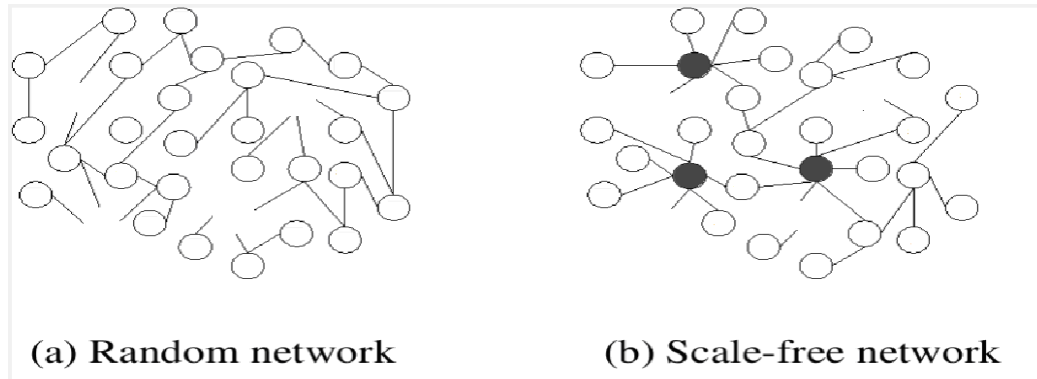


Figure (2.3): Difference (1) between Random and Scale-Free Network.

In Figure (2.4) if we cut the same nodes from random and scale-free network, random network split to more components than scale-free network.

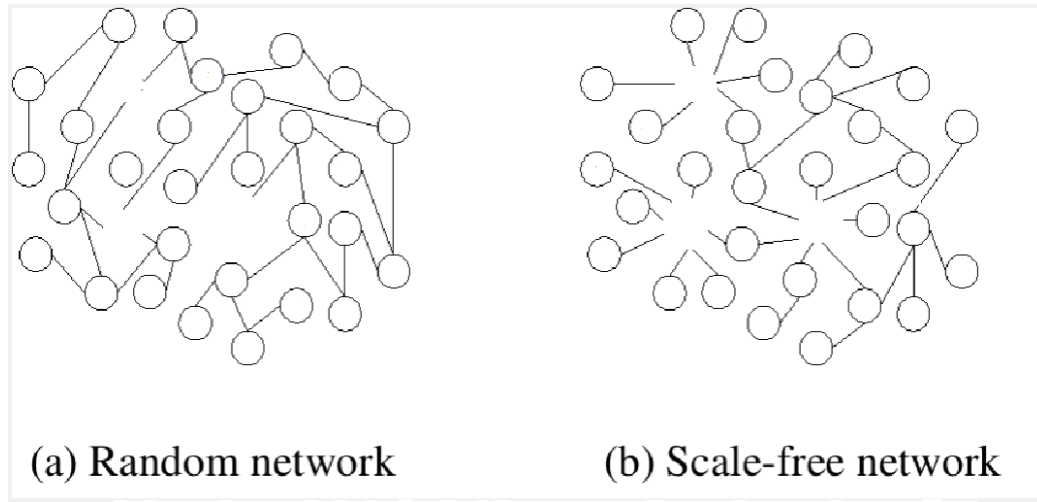
2- Scale-free networks are more vulnerable against non-random attacks [3]:

If we remove nodes that have the highest degree, the network will be quickly disintegrated.

Because in scale-free network the nodes that have high degree are adjacent to 70% of all nodes, then in removing these nodes, the network will break down into many components (the network will be disconnected).

Example (2.2)

Consider the Network in Example (2.1)



Figure(2.4): Difference(2) Between Random and Scale-Free Network.

After removing central nodes we find that the scale-free network becomes disconnected but a random network stays connected.

3- A scale-free network has a shorter average path length than that in a random network [3]:

Example (2.3)

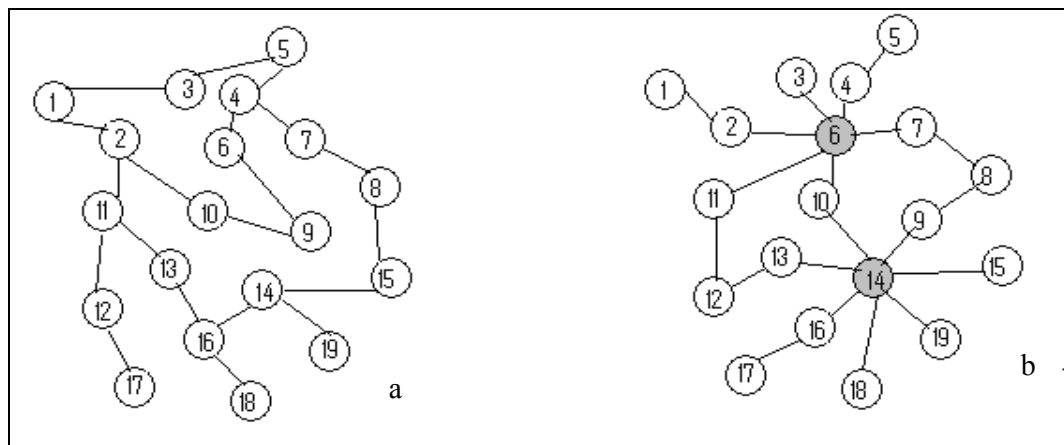


Figure (2.5): Different (3) Between Random and Scale-Free Network.

To compute the shortest path between each pair of nodes we use Floyd's algorithm.

Floyd's algorithm computes the shortest path between any two nodes in the graph in weighted graph. Here we use this algorithm to compute the shortest between each pair of nodes in unweighted graph, by giving each edge weight to one.

Algorithm (2.1): Floyd's algorithm [17]

```

    if  $i=j$  then  $L_{ij}(0)=0$ 

    If nodes  $i$  and  $j$  are adjacent  $L_{ij}(0)=\text{length of edge } ij$ 

    else  $L_{ij}(0) = \infty$ 

    for  $k=1:n$ 

    for  $i=1:n$ 

    for  $j=1:n$ 

     $L_{ij}(k+1) = \min(L_{ij}(k) , L_{ik}(k)+L_{kj}(k))$ 

    End

    End

    End

```

```
S=L(n)
```

```
/* S : the array of shortest path s.t. S(i,j)=shortest path between nodes i and j */
```

To compute the average shortest path we use the following algorithm

Algorithm (2.2): Compute Average Shortest Path

```
Sum=0;
```

```
for i=1:n
```

```
    for j=1:n
```

```
        sum=sum+S(i,j);
```

```
    end
```

```
end
```

```
Average=sum/(n*n);
```

S_1 is the length of the shortest path for the random network in Figure (2.6) showed in the following matrix:

$$S_1 = \begin{bmatrix} 0 & 1 & 1 & 3 & 2 & 4 & 4 & 5 & 3 & 2 & 2 & 3 & 3 & 5 & 6 & 4 & 4 & 5 & 6 \\ 1 & 0 & 2 & 4 & 3 & 3 & 5 & 6 & 2 & 1 & 1 & 2 & 2 & 4 & 5 & 3 & 3 & 4 & 5 \\ 1 & 2 & 0 & 2 & 1 & 3 & 3 & 4 & 4 & 3 & 3 & 4 & 4 & 6 & 5 & 5 & 5 & 6 & 7 \\ 3 & 4 & 2 & 0 & 1 & 1 & 1 & 2 & 2 & 3 & 5 & 6 & 6 & 4 & 3 & 5 & 7 & 6 & 5 \\ 2 & 3 & 1 & 1 & 0 & 2 & 2 & 3 & 3 & 4 & 4 & 5 & 5 & 5 & 4 & 6 & 6 & 7 & 6 \\ 4 & 3 & 3 & 1 & 2 & 0 & 2 & 3 & 1 & 2 & 4 & 5 & 5 & 5 & 4 & 6 & 6 & 7 & 6 \\ 4 & 5 & 3 & 1 & 2 & 2 & 0 & 1 & 3 & 4 & 6 & 7 & 5 & 3 & 2 & 4 & 8 & 5 & 4 \\ 5 & 6 & 4 & 2 & 3 & 3 & 1 & 0 & 4 & 5 & 5 & 6 & 4 & 2 & 1 & 3 & 7 & 4 & 3 \\ 3 & 2 & 4 & 2 & 3 & 1 & 3 & 4 & 0 & 1 & 3 & 4 & 4 & 6 & 5 & 5 & 5 & 6 & 7 \\ 2 & 1 & 3 & 3 & 4 & 2 & 4 & 5 & 1 & 0 & 2 & 3 & 3 & 5 & 6 & 4 & 4 & 5 & 6 \\ 2 & 1 & 3 & 5 & 4 & 4 & 6 & 5 & 3 & 2 & 0 & 1 & 1 & 3 & 4 & 2 & 2 & 3 & 4 \\ 3 & 2 & 4 & 6 & 5 & 5 & 7 & 6 & 5 & 3 & 1 & 0 & 2 & 4 & 5 & 3 & 1 & 4 & 5 \\ 3 & 2 & 4 & 6 & 5 & 5 & 5 & 4 & 4 & 3 & 1 & 2 & 0 & 2 & 3 & 1 & 3 & 2 & 3 \\ 5 & 4 & 6 & 4 & 5 & 5 & 3 & 2 & 6 & 5 & 3 & 4 & 2 & 0 & 1 & 1 & 5 & 2 & 1 \\ 6 & 5 & 5 & 3 & 4 & 4 & 2 & 1 & 5 & 6 & 4 & 5 & 3 & 1 & 0 & 2 & 6 & 3 & 2 \\ 4 & 3 & 5 & 5 & 6 & 6 & 4 & 3 & 5 & 4 & 2 & 3 & 1 & 1 & 2 & 0 & 4 & 1 & 2 \\ 4 & 3 & 5 & 7 & 6 & 6 & 8 & 7 & 5 & 4 & 2 & 1 & 3 & 5 & 6 & 4 & 0 & 5 & 6 \\ 5 & 4 & 6 & 6 & 7 & 7 & 5 & 4 & 6 & 5 & 3 & 4 & 2 & 2 & 3 & 1 & 5 & 0 & 3 \\ 6 & 5 & 7 & 5 & 6 & 6 & 4 & 3 & 7 & 6 & 4 & 5 & 3 & 1 & 2 & 2 & 6 & 3 & 0 \end{bmatrix}$$

Average shortest path between nodes in the random network = 3.55

S_2 is the length of the shortest path for the scale-free network in Figure

(2.6) showed in the following matrix:

$$S_2 = \begin{bmatrix} 0 & 1 & 3 & 3 & 4 & 2 & 3 & 4 & 5 & 3 & 3 & 4 & 5 & 4 & 5 & 5 & 6 & 5 & 5 \\ 1 & 0 & 2 & 2 & 3 & 1 & 2 & 3 & 4 & 2 & 2 & 3 & 4 & 3 & 4 & 4 & 5 & 4 & 4 \\ 3 & 2 & 0 & 2 & 3 & 1 & 2 & 3 & 4 & 2 & 2 & 3 & 4 & 3 & 4 & 4 & 5 & 4 & 4 \\ 3 & 2 & 2 & 0 & 1 & 1 & 2 & 3 & 4 & 2 & 2 & 3 & 4 & 3 & 4 & 4 & 5 & 4 & 4 \\ 4 & 3 & 3 & 1 & 0 & 2 & 3 & 4 & 5 & 3 & 3 & 4 & 5 & 4 & 5 & 5 & 6 & 5 & 5 \\ 2 & 1 & 1 & 1 & 2 & 0 & 1 & 2 & 3 & 1 & 1 & 2 & 3 & 2 & 3 & 3 & 4 & 3 & 3 \\ 3 & 2 & 2 & 2 & 3 & 1 & 0 & 1 & 2 & 2 & 2 & 3 & 4 & 3 & 4 & 4 & 5 & 4 & 4 \\ 4 & 3 & 3 & 3 & 4 & 2 & 1 & 0 & 1 & 3 & 3 & 4 & 3 & 2 & 3 & 3 & 4 & 3 & 3 \\ 5 & 4 & 4 & 4 & 5 & 3 & 2 & 1 & 0 & 2 & 4 & 3 & 2 & 1 & 2 & 2 & 3 & 2 & 2 \\ 3 & 2 & 2 & 2 & 3 & 1 & 2 & 3 & 2 & 0 & 2 & 3 & 2 & 1 & 2 & 2 & 3 & 2 & 2 \\ 3 & 2 & 2 & 2 & 3 & 1 & 2 & 3 & 3 & 2 & 0 & 1 & 3 & 2 & 3 & 3 & 4 & 3 & 3 \\ 5 & 4 & 4 & 4 & 5 & 3 & 4 & 3 & 2 & 2 & 4 & 1 & 2 & 1 & 2 & 2 & 3 & 2 & 2 \\ 5 & 4 & 4 & 4 & 5 & 3 & 4 & 3 & 2 & 2 & 4 & 1 & 0 & 1 & 2 & 2 & 3 & 2 & 2 \\ 4 & 3 & 3 & 3 & 4 & 2 & 3 & 2 & 1 & 1 & 3 & 2 & 1 & 0 & 1 & 1 & 2 & 1 & 1 \\ 5 & 4 & 4 & 4 & 5 & 3 & 4 & 3 & 2 & 2 & 4 & 3 & 2 & 1 & 0 & 2 & 3 & 2 & 2 \\ 5 & 4 & 4 & 4 & 5 & 3 & 4 & 3 & 2 & 2 & 4 & 3 & 2 & 1 & 2 & 0 & 1 & 2 & 2 \\ 6 & 5 & 5 & 5 & 6 & 4 & 5 & 4 & 3 & 3 & 5 & 4 & 3 & 2 & 3 & 1 & 0 & 3 & 3 \\ 5 & 4 & 4 & 4 & 5 & 3 & 4 & 3 & 2 & 2 & 4 & 3 & 2 & 1 & 2 & 2 & 3 & 0 & 2 \\ 5 & 4 & 4 & 4 & 5 & 3 & 4 & 3 & 2 & 2 & 4 & 3 & 2 & 1 & 2 & 2 & 3 & 2 & 0 \end{bmatrix}$$

The average shortest path between nodes = 2.80

2.2 Power Law Distribution

Power Law distribution is a polynomial relationship that exhibits the property of scale invariance.

$$f(x) = ax^k + O(x^k) \quad (2.1)$$

Where a, k are constant, and $O(x^k)$ is an asymptotically small function of x . [20]

of node's degree is characterized by the degree distribution $P(K)$ which gives the probability that a randomly selected node has exactly k edges .

Albert-Laszlo Barabasi, Zoltan Dezso, Erazsebt Ravasz, Soon-Hyung Yook and Zoltan Oltv reach to a result that for most large networks, including the World-Wide Web, Internet, metabolic and protein networks, language or sexual networks, the degree distribution follows a power-law distribution [6]:

$$P(k) \approx k^{-\gamma}$$

$P(k) \approx k^{-\gamma_{in}}$, where γ_{in} is different from one graph to another.

$P(k) \approx k^{-\gamma_{out}}$, where γ_{out} is different from graph to another.

In the following table we can see the scaling exponents characterizing the degree distribution of several scale-free networks, for which $P(k)$ follows a power-law . We indicate the size of the network and its average

degree by $\langle K \rangle$ For directed networks we list separately the in-degree (γ_{in}) and out-degree (γ_{out}) exponents, while for the undirected networks, marked with a star [6]

TABLE (2.1): Values of γ_{in} , γ_{out} and $\langle K \rangle$ For Some Scale-Free

Networks. [6]

Network	Size	$\langle K \rangle$	γ_{in}	γ_{out}
WWW	325,729	4.51	2.45	2.1
WWW	4×10^7	7	2.38	2.1
WWW	2×10^8	7.5	2.72	2.1
Internet , domain *	3,015- 4,389	3.42- 3.76	2.1- 2.2	2.1- 2.2
Internet , router *	3,888	2.57	2.48	2.48
Internet , router *	150,000	2.66	2.4	2.4
Movie actors *	212,250	28.78	2.3	2.3
Coauthors ,SPIRES *	56,627	173	1.2	1.2
Coauthors , neurol. *	209,293	11.54	2.1	2.1
Coauthor , math *	70,975	3.9	2.5	2.5
Metabolic, E.coli	778	7.4	2.2	2.2
Protein, S. cerev.*	1870	2.39	2.4	2.4
Ythan estuary *	134	8.7	1.05	1.05
Silwood park *	154	4.75	1.13	1.13

Citation	783,339	8.57		3
Phone-call	53X106	3.16	2.1	2.1
Words, concurrence!	460,902	70.13	2.7	2.7
Words, synonyms!	22,311	13.48	2.8	2.8
Protein, S. Cerev*	9,85	1.83	2.5	2.5
Comic Book Characters	6,486	14.9	0.66	3.12
E-mail	59,912	2.88	2.03	1.49
Protein Domains*	876	9.32	1.6	1.6
Prot. Dom. (PromDom)*	5995	2.33	2.5	2.5
Prot. Dom. (Pform)*	2478	1.12	1.7	1.7
Prot. Dom. (Prosite)*	13.60	0.77	1.7	1.7

From this table we can see clearly that values of γ_{in} , γ_{out} and $\langle K \rangle$ depend on the type and size of network.

2.3 Clustering

2.3.1 Definition:

Clustering is grouping similar data items together [15].

Clustering motivation:

- 1- To provide automated tools to help in constructing categories or taxonomies [15].
- 2- To minimize the effects of human factors in the process [15].

2.3.2 Clustering Methods

There are many different methods (algorithms) for clustering. These may be divided into two basic types:

- 1- Hierarchical clustering.
- 2- Partitioned clustering.

2.3.2.1 Hierarchical Clustering

Hierarchical clustering Proceeds successively by either merging small clustering into larger ones or splitting larger clustering [15]. We can divide hierarchical clustering into two main type: Agglomerative method, and Divisive method [12][9].

- I- Agglomerative hierarchical method: clusters are successively merged until one cluster remains [9].

There are many different Agglomerative method of clustering. The main difference between them is in how to compute the distance between any two clusters. Some of these methods are:

- a- Single linkage method: The distance between two clusters is based on the points in each cluster that are nearest together[9].

$$D_{KL} = \min_{\substack{i \in C_k \\ j \in C_L}} d(x_i, x_j) \quad (2.2)$$

- b- Complete linkage method: The distance between two clusters is based on the points in each cluster that are furthest apart[9].

$$D_{KL} = \max_{\substack{i \in C_k \\ j \in C_L}} d(x_i, x_j) \quad (2.3)$$

- c- Centroid linkage method: The distance between clusters is defined as the (squared) Euclidean distance between cluster centroids \bar{X}_K and \bar{X}_L ^[11].

$$D_{KL} = \|\bar{X}_K - \bar{X}_L\|^2 \quad (2.4)$$

Where \bar{X}_K is the center of subgraph K and \bar{X}_L is the center of subgraph L.

- d- Average linkage method: The distance between clusters is the average distance between pairs of observations ^[11].

$$D_{KL} = \frac{1}{n_L n_K} \sum_{i \in C_K} \sum_{j \in C_L} d(x_i, x_j) \quad (2.5)$$

Where n_L is the number of nodes in subgraph L, and n_K is the number of nodes in subgraph K.

Algorithm(2.3): Agglomerative algorithm (single linkage method)[16]

1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.
2. Find the least dissimilar pair of clusters in the current clustering, say pair (r), (s), according to

$$d[(r),(s)] = \min d[(i),(j)]$$

where the minimum is over all pairs of clusters in the current clustering.

3. Increment the sequence number : $m = m + 1$. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to

$$L(m) = d[(r),(s)]$$

4. Update the proximity matrix, D , by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined in this way:

$$d[(k), (r,s)] = \min d[(k),(r)], d[(k),(s)]$$

5. If all objects are in one cluster, stop. Else, go to step 2.

Example (2.4): [16]

We apply agglomerative algorithm (single linkage method) to cluster some Italian cities. The distances in kilometers between these cities given in the matrix.



Figure (2.6): Map of Italian Cities.

	B	FI	M	N	R	T
A			I	A	M	O
B	0	6	8	2	4	9

A		62	77	55	12	96
F	6		2	4	2	4
I	62	0	95	68	68	00

25

M	8	2		7	5	1
I	77	95	0	54	64	38
N	2	4	7		2	8
A	55	68	54	0	19	69
R	4	2	5	2		6
M	12	68	64	19	0	69
T	9	4	1	8	6	
O	96	00	38	69	69	0

In step 0 we have 6 items each item has only one component, we called each item cluster.

The matrices show the distance in kilometer between the cities. Form the adjacency matrix we find that the closest pair of clusters is MI and TO.

In step 1 we cluster MI and TO into one item the distance between the new item and the other items = minimum distance between MI or TO and any other nodes.



Figure (2.7): Step 1 We Cluster MI and TO in One Item.

	BA	FI	MI/TO	NA	RM
BA	0	66 2	877	25 5	41 2
FI	66 2	0	295	46 8	26 8
MI/TO	87 7	29 5	0	75 4	56 4

26

NA	255	468	754	0	219
RM	412	268	564	219	0

In step 2 the closest pair of clusters is NA and RM, we cluster them in one cluster.



Figure (2.8): In Step 2 We Cluster RM and NA In to One Item.

	B A	FI	MI/T O	NA/R M
BA	0	6 62	877	255
FI	6 62	0	295	268
MI/T O	8 77	2 95	0	564
NA/R M	2 55	2 68	564	0

In step 3 the closest pair of clusters is BA and NA\RM



Figure (2.9): Step3 We Cluster BA With NA\RM Into One Item.

	BA/ NA/RM	FI	MI/T O
BA /NA/RM	0	2 68	564
FI	268	0	295
MI/TO	564	2 95	0

In level 4 the closest pair of clusters is BA\ NA\RM and FI



Figure (2.10): In Step 4 We Cluster FI With BA\NA\RM.

	BA/FI/ NA/RM	MI/T O
BA/FI/ NA/RM	0	295
MI/TO	295	0

28

In the last step we have two clusters, which will be clustered into one cluster.

II - Divisive hierarchical methods: begin with all objects in one cluster.

Groups are continually divided until there are many clusters [9].

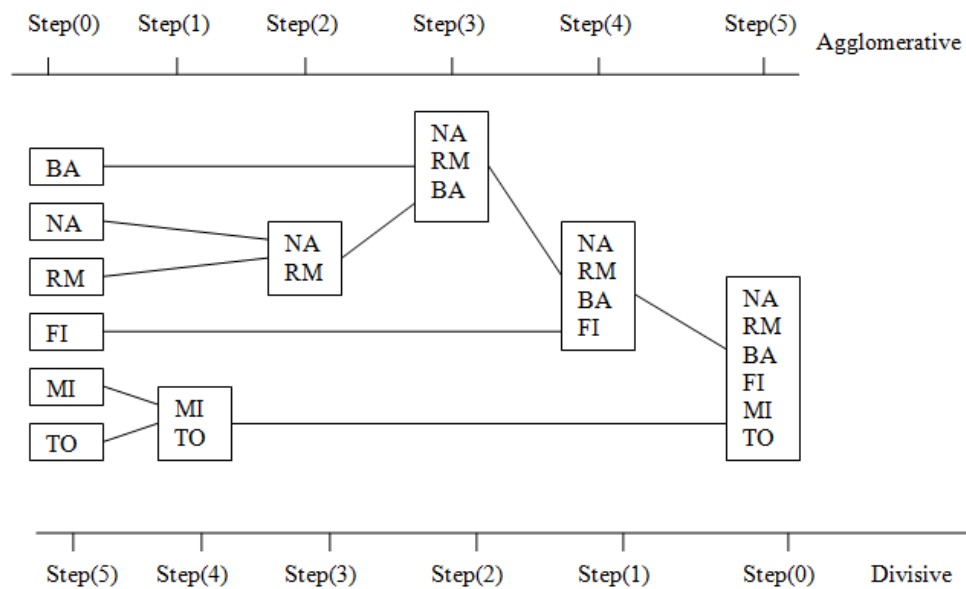


Figure (2.11): Steps of Hierarchal Clustering Using Agglomerative and Divisive Method.

If we apply Divisive method on the graph that we explain in Example (2.4) we start considering all the cities in one cluster then start dividing the graph until we each city become a cluster.

2.3.2.2 Partitioned Clustering

Partitioned clustering, attempts to directly decompose the data set into a set of disjoint clusters [15].

29

K-means clustering method is a nonhierarchical clustering method , which aims to partition n observation into k clusters in which each observation belongs to the cluster with the nearest center [4][21] .

There are several variants of the k-means clustering algorithm, but most variants involve an iterative scheme that operates over a fixed number of clusters, while attempting to satisfy the following properties[5]:

1. Each class has a center which is the mean position of all the samples in that class.
2. Each sample is in class whose center is closest to.

Algorithm (2.4): Main Algorithm in k-Means Clustering [1]

1. *Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.*
2. *Assign each object to the group that has the closest centroid.*
3. *When all objects have been assigned, recalculate the positions of the K centroids.*
4. *Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric*

to be minimized can be calculated.

30

Example (2.5)

Cluster the following point into 3 clusters

{A1(3,5) , A2(5,2) , A3(1,7) , A4(12,1) , A5(10,1), A6(5,11), A7(4,4),
A8(7,10), A9(9,12), A10(10,3)}.

Where the distance between 2 nodes is :

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Iteration 1

We choose 3 nodes in random way and consider each point as a center of cluster. Suppose that A1,A3, and A10 are center.

Points	Dist mean1 (3,5)	Dist mean 2 (1,7)	Dist mean 3 (10,3)	Cluste r
A1(3.5)	0	2.8284	7.2801	1
A2(5.2)	3.6056	6.4031	5.0990	1
A3(1,7))	2.8284	0	9.8489	2

A4(12,1)	9.8489	12.5300	2.8284	3
A5(10,1)	8.0623	10.8167	2.0000	3
A6(5,11)	6.3246	5.6569	9.4340	2
A7(4,4)	1.4142	4.2426	6.0828	1
A8(7,10)	6.4031	6.7082	7.6158	1

31

A9(9,12)	9.2195	9.4340	9.0554	3
A10(10,3)	7.2801	9.8489	0	3

cluster 1 contain nodes : A1 (3,5) ,A 2 (5,2) , A7 (4,4) , and A8 (7,10)

cluster 2 contain nodes : A3 (1,7) , and A6 (5,11)

cluster 3 contain nodes: A4(12,1), A5(10,1) ,A9(9,12) ,and A10 (10,3)

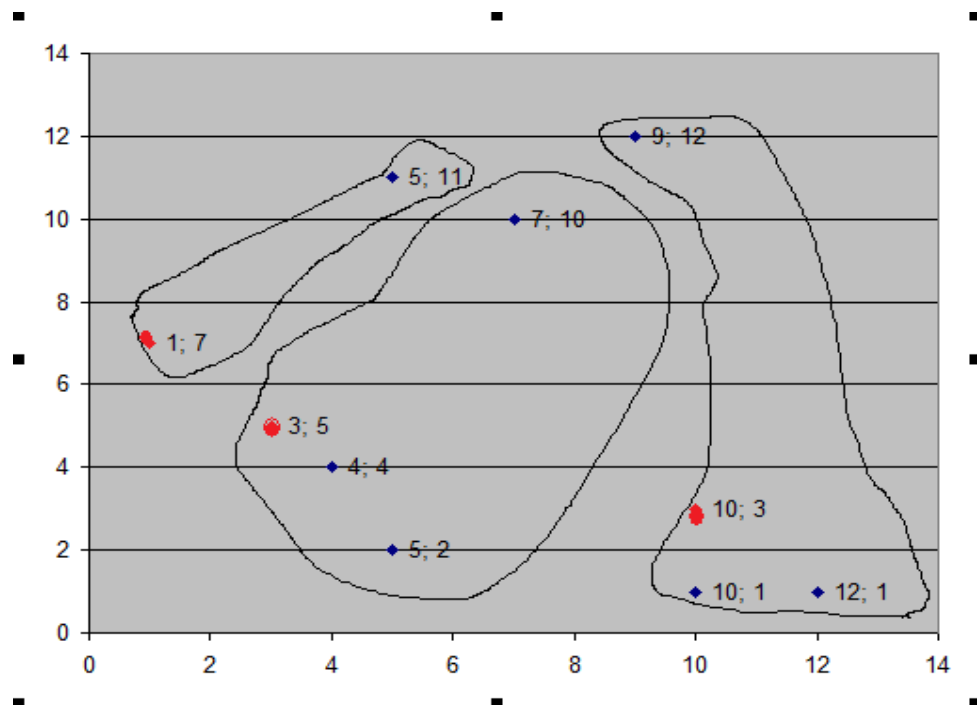


Figure (2.12): Graph of the Nodes IN Example (2.5), Nodes Colored in Red are the Centers of the Cluster.

center of cluster 1 $=((3+5+4+7)/4, (5+2+4+10)/4)=(4.75, 5.25)$

center of cluster 2 $=((1+5)/2, (7+11)/2) = (3, 9)$

center of cluster 3 $=((12+10+9+10)/4, (1+1+12+3)/4) = (10.25, 4.25)$

32

Iteration 2

Points	Dist mean 1 (4.75,5.25)	Dist mean2 (3,9)	Dist mean 3 (10.25,4.25)	Cluster
A1(3,5)	1.7678	4	7.2887	1
A2(5,2)	3.2596	7.2801	5.7118	1
A3(1,7))	4.1382	2.8284	9.6501	2
A4(12,1)	8.4039	12.0416	3.6912	3
A5(10,1)	6.7546	10.6301	3.2596	3
A6(5,11)	5.7554	2.8284	8.5513	2
A7(4,4)	1.4577	5.0990	6.2550	1
A8(7,10)	5.2559	4.1231	6.6049	2
A9(9,12)	7.9765	6.7082	7.8502	2
A10(10,3)	5.7118	9.2195	1.2748	3

cluster 1 contain nodes : A1 (3,5) ,A 2 (5,2) ,and A7 (4,4)

cluster 2 contain nodes: A3(1,7), A6 (5,11), A8 (7,10) ,and A9 (9,12)

cluster 3 contain nodes : A4 (12,1) , A5 (10,1),and A10 (10,3)

33

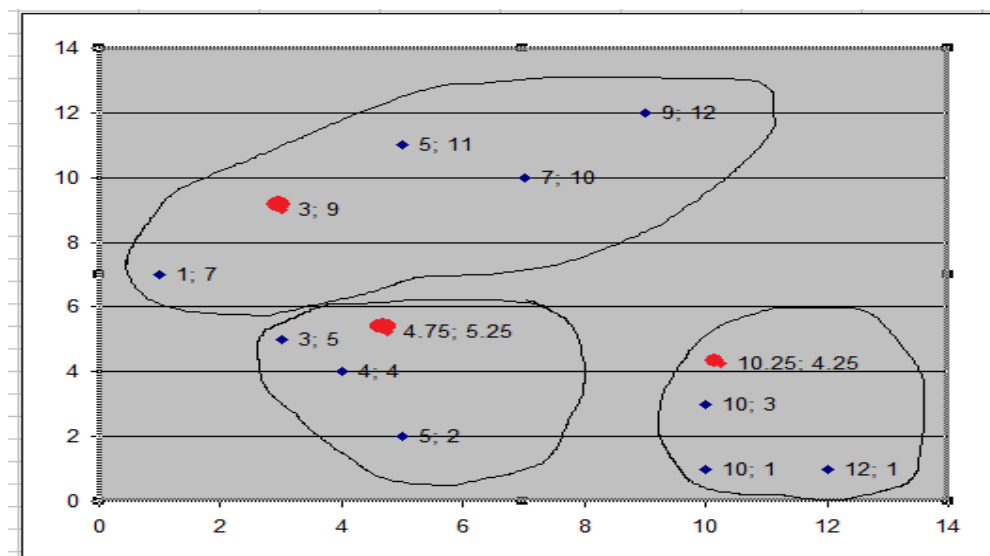


Figure (2.13): Clusters in Iteration Two, the Point Colored in Red are the Centers of the Clusters.

center of cluster 1 = (4 , 3.6667)

center of cluster 2 = (5.5 , 10)

center of cluster 3 = (10.6667 , 1.6667)

Iteration 3

Points	Dist mean 1 (4,3.6667)	Dist mean2 (5.5 , 10)	Dist mean 3 (10.6667,1.6667)	Cluster
A1(3.5)	1.6667	5.5902	8.3600	1
A2(5.2)	1.9437	8.0156	5.6765	1
A3(1,7))	4.4845	5.4083	11.0403	1
A4(12,1)	8.4327	11.1018	1.4907	3

34

A5(10,1)	6.5659	10.0623	0.9428	3
A6(5,11)	7.4012	1.1180	10.9189	2
A7(4,4)	0.3333	6.1847	7.0632	1
A8(7,10)	7.0079	1.5000	9.1043	2
A9(9,12)	9.7183	4.0311	10.4669	2
A10(10,3)	6.0369	8.3217	1.4907	3

cluster 1 contain nodes : A1 (3,5) , A2 (5,2) , A3(1,7), and A7 (4,4)

cluster 2 contain nodes : A6 (5,11) , A8 (7,10) , and A9(9,12)

cluster 3 contain nodes : A4 (12,1) , A5 (10,1) , and A10 (10,3)

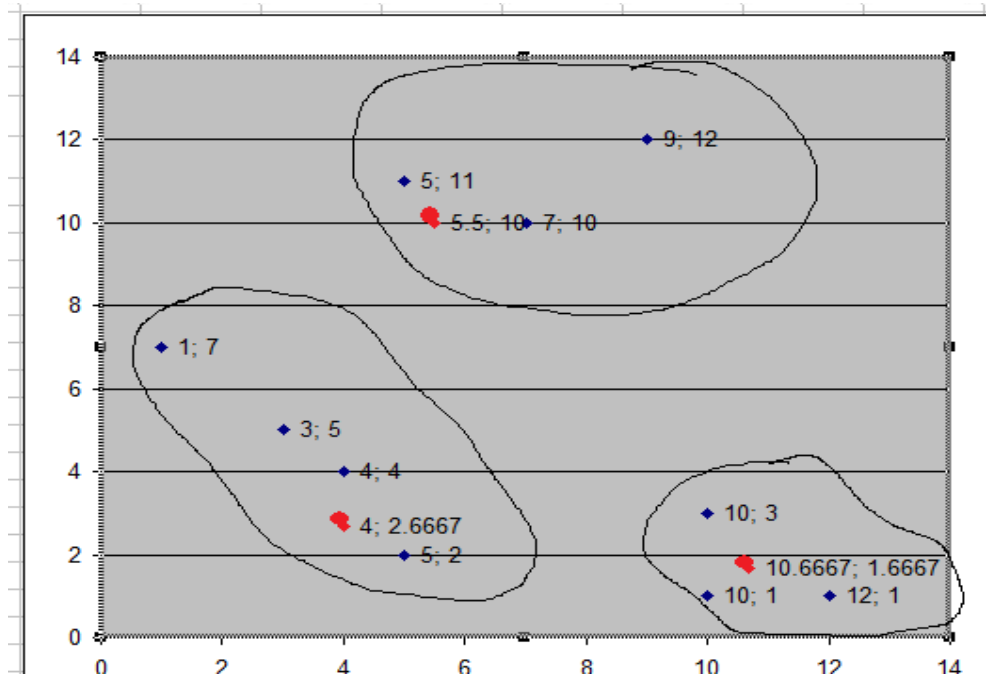


Figure (2.14): Cluster of Iteration Three.

35

center of cluster 1 = (3.25 , 4.5)

center of cluster 2 is (7 , 11)

center of cluster 3 is (10.6667 , 1.6667)

Iteration 4

Points	Dist mean1 (3.25 , 4.5)	Dist mean 2 (7 , 11)	Dist mean 3 (10.6667 , .6667)	Cluste r
A1(3.5)	0.5590	7.2111	8.3600	1
A2(5.2)	3.0516	9.2195	5.6765	1

A3(1,7))	3.3634	7.2111	11.0403	1
A4(12,1)	9.4240	11.1803	1.4907	3
A5(10,1)	7.6035	10.4403	0.9428	3
A6(5,11)	6.7315	2	10.9189	2
A7(4,4)	0.9014	7.6158	7.0632	1
A8(7,10)	6.6568	1	9.1043	2
A9(9,12)	9.4505	2.2361	10.4669	2
A10(10,3)	6.9147	8.5440	1.4907	3

cluster 1 contain nodes : A1 (3,5) , A2 (5,2) , A3 (1,7), and A7 (4,4)

cluster 2 contain nodes : A6 (5,11) , A8 (7,10) , and A9 (9,12)

cluster 3 contain nodes : A4 (12,1) , A5 (10,1) , A10 (10,3)

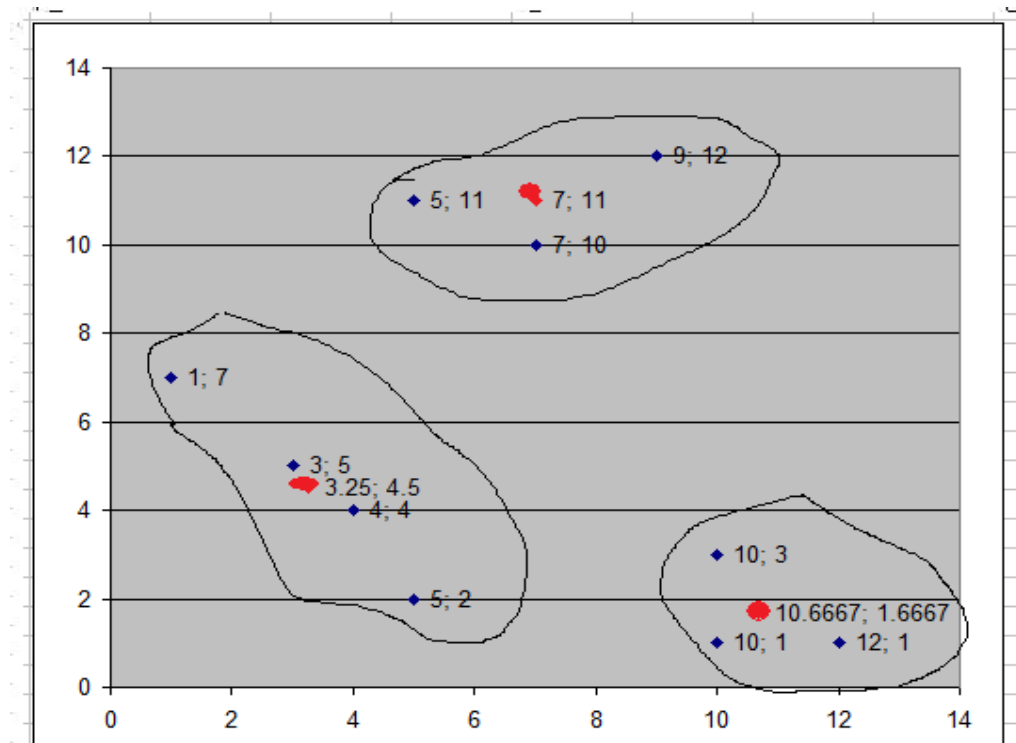


Figure (2.15): Cluster of Iteration Four.

We repeat the same iteration until no new centers for the cluster appear.

In this example we reach to the following three clusters:

Cluster 1 contain nodes: A1(3,5), A2(5,2), A3(1,7) ,and A7 (4,4) . Center of cluster 1 is: (3.25 , 4.5).

Cluster 2 contain nodes : A6 (5,11) , A8 (7,10) , and A9 (9,12).

Center of cluster two is: (7 , 11) .

Cluster 3 contain nodes : A4 (12,1) , A5 (10,1) , andA10 (10,3) .

Center of cluster three is: (10.667 , 1.667).

Chapter Three

Bridging Centrality

3.1 Bridging Centrality

A bridging node is a node located between modules. It is a node that connects densely connected components in a graph. The bridging nodes in a graph are identified on the basis of their high value of bridging centrality relative to other nodes in the same graph (or a concept and formula to identify bridging nodes in scale-free networks). Bridging centrality of a node measures the global and local features of a node, respectively. Bridging centrality of node v , $(C_R(v))$, is the product of the betweenness centrality of a node v , $(C_B(v))$, and the bridging coefficient of a node v , $(BC(v))$. [13]

$$C_R(V) = BC(V) * C_B(V) \quad (3.1)$$

To evaluate Bridging Centrality, we need to evaluate the bridging coefficient (BC) and betweenness centrality (C_B).

3.2 Bridging Coefficient

The bridging coefficient of a node determines the extent of how well the node is located between high degree nodes.

$$BC(V) = \frac{D(v)^{-1}}{\sum_{i \in N(v)} D(i)^{-1}} \quad (3.2)$$

The symbol $D(v)$ denotes the degree of node v (number of nodes directly connected to node v), and $N(v)$ denotes the neighbors of node v

(is the set of nodes that are adjacent to node v). [13] To evaluate bridging coefficient, we need to find the degree of each node. From the adjacency matrix we can find the matrix Ne which is $(n, 1)$ matrix where $Ne(i,1) = \deg(i)$. Then from A and Ne we find the vector BC , where $BC(i)$ equal the bridging coefficient for node i .

We write the following algorithm to compute $BC(v)$ for $v \in V$.

Algorithm (3.1): BC Algorithm.

```
function [bc]
//first we find Ne for the graph
for i=1:n
    Ne(i)=0;
    for j=1:n
        if A(i,j)==1;
            Ne(i)=Ne(i)+1;
        end
    end
end
//finding BC for each node
for i=1:n
    t=0;
    for j=1:n
        if A(i,j)==1
            t=t+1/Ne(j);
        end
    end
end
```

```

end

BC(i)=(1/Ne(i))/t;

End

```

3.3 Betweenness Centrality

The betweenness centrality is a measure of the global importance of a node that assesses the proportion of a shortest path between all node pairs that pass through the node of interest.

$$C_B = \sum_{\substack{s \neq t \neq v \\ s, t, v \in V}} \frac{\sigma_{st}(v)}{\sigma_{st}} \bigg/ ((n-1) * (n-2)) \quad (3.3)$$

Where σ_{st} denotes the number of shortest path from s to t and $\sigma_{st}(v)$ denotes number of shortest paths from s to t that pass through node v .[13]

$$\sigma_{st}(V) = \begin{cases} 0 & d_G(s, t) < d_G(s, v) + d_G(v, t) \\ \sigma_{sv} * \sigma_{vt} & otherwise \end{cases} \quad (3.4)$$

The number $(n-1)*(n-2)$ gives the number of pairs in the graph excluding vertex v .

Betweenness Centrality is important in the analysis of social network but is costly to compute. The following algorithm is called "The faster algorithm" written by Ulrik Brandes and is applied to compute C_B in unweighted undirected graph.

The algorithm is [7]

Algorithm (3.2): Faster Algorithm to Find C_B

```
 $C_B[v]=0$  ,  $v \in V$ ;  
for  $s \in V$  do  
    S is empty stack;  
    P[w] is empty list ,  $w \in V$ ;  
     $\sigma[t]=0$  ,  $t \in V$  ;  $\sigma[s]=1$ ;  
     $d[t]=-1$  ,  $t \in V$  ;  $d[s]=0$ ;  
    Q is empty queue ;  
    Enqueue (s,Q);  
    while Q not empty do  
        dequeue(v,Q);  
        push (v,S);  
        for  $w \in N(v)$  do  
            if  $d[w]<0$  then  
                enqueue(w,Q);  
                 $d[w]=d[v]+1$ ;  
            end  
            if  $d[w]=d[v]+1$  then  
                 $\sigma[w]=\sigma[w]+\sigma[v]$ ;  
                append(v,P[w])  
            end  
        end  
    end  
end  
 $\delta[v]=0$  ,  $v \in V$ ;
```

```

while S not empty do
    pop(w,S);
    for v ∈ P[w] do
         $\delta[v] = \delta[v] + (\sigma[v] / \sigma[w]) * (1 + \delta[w]);$ 
    end
    if w ≠ s then
         $C_B[w] = C_B[w] + \delta[w];$ 
         $\delta[w]$ 
    end
end
end

```

After deep study of this subject, we reached to another algorithm that depends on two main matrices. The first of which is the matrix (S1) where $S1(i,j)$ = number of shortest path from node i to node j, ($i, j \in V$) and the second matrix is (S2) where $S2(i,j)$ = length of shortest paths from node i to j, ($i, j \in V$).

Algorithm (3.3): C_B Algorithm

```

function [cb]
compute Num % this array contain number of shortest path between
               any two nodes we assume that this array was computed%
compute L % this array contain length of shortest path between
               any two nodes we assume that this array was computed%
//calculate Bridging Centrality ( $C_B$ )
for i=1:n

```

```

s=0;
for j=1:n
    if j~=i
        for k=1:n
            if k~=i
                if L(j,k)<L(j,i)+L(i,k)
                    t=0;
                else
                    t=Num(j,i)*Num(i,k)/Num(j,k);
                end
                s=s+t;
            end
        end
    end
end

CB(i,1)=s/((n-1)*(n-2));

End

```

Our algorithm is easier to understand by any average person with little information about programming as a math student. But Ulrik Brandes algorithm needs deep understanding of data structure to understand.

Example (3.1):

We apply our Algorithms Algorithm(3.1) to compute BC and Algorithm(3.3) to compute C_R on the graph in Figure(3.1).

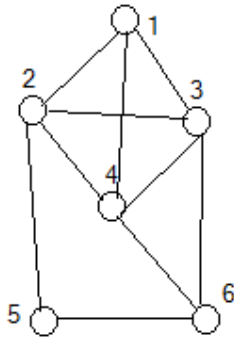


Figure (3.1): Simple Graph

Here we show how we compute BC and C_R for nodes 2 and 3 :

First (BC)

To compute BC we need the adjacency matrix and the degree of each node,

the adjacency matrix is:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix},$$

The vector Ne contain the degree of each node, the vector is:

$$Ne = \begin{bmatrix} 3 \\ 4 \\ 4 \\ 4 \\ 2 \\ 3 \end{bmatrix}$$

$n=6$, number of nodes

$i = 2 \quad t = 0$

$j = 1 \quad A(2,1) = 1 \quad t = 0 + \frac{1}{3}$

$j = 2 \quad A(2,2) = 0$

$$\begin{aligned}
j=3 \quad A(2,3) &= 1 \quad t = \frac{1}{3} + \frac{1}{4} \\
j=4 \quad A(2,4) &= 1 \quad t = \frac{1}{3} + \frac{1}{4} + \frac{1}{4} \\
j=5 \quad A(2,5) &= 1 \quad t = \frac{1}{3} + \frac{1}{4} + \frac{1}{4} + \frac{1}{2} \\
j=6 \quad A(2,6) &= 0 \\
t &= \frac{4}{3}
\end{aligned}$$

$$BC(2) = \frac{(\frac{1}{Ne(2)})}{t} = \frac{1}{4} * \frac{3}{4} = 0.1875$$

$$\begin{aligned}
i=3 \quad t &= 0 \\
j=1 \quad A(3,1) &= 1 \quad t = 0 + \frac{1}{3} \\
j=2 \quad A(3,2) &= 1 \quad t = \frac{1}{3} + \frac{1}{4} \\
j=3 \quad A(3,3) &= 0 \\
j=4 \quad A(3,4) &= 1 \quad t = \frac{1}{3} + \frac{1}{4} + \frac{1}{4} \\
j=5 \quad A(3,5) &= 0 \\
j=6 \quad A(3,6) &= 1 \quad t = \frac{1}{3} + \frac{1}{4} + \frac{1}{4} + \frac{1}{3} \\
t &= \frac{7}{6}
\end{aligned}$$

$$BC(3) = \frac{(\frac{1}{Ne(3)})}{t} = \frac{1}{4} * \frac{6}{7} = 0.214286$$

We compute BC for the other way in the same way

Second (C_B):

To compute C_B we need two arrays: One- (Num) this array contain number of shortest path between any two nodes. Two- (L) this array contain length of shortest path between any two nodes.

$$Num = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 2 \\ 1 & 0 & 1 & 1 & 1 & 3 \\ 1 & 1 & 0 & 1 & 2 & 1 \\ 1 & 1 & 1 & 0 & 2 & 1 \\ 1 & 1 & 2 & 2 & 0 & 1 \\ 2 & 3 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} 0 & 1 & 1 & 1 & 2 & 2 \\ 1 & 0 & 1 & 1 & 1 & 2 \\ 1 & 1 & 0 & 1 & 2 & 1 \\ 1 & 1 & 1 & 0 & 2 & 1 \\ 2 & 1 & 2 & 2 & 0 & 1 \\ 2 & 2 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Here we will compute C_R for nodes 2 and 3:

$$\begin{aligned}
i &= 2 & s &= 0 \\
j &= 1 & k &= 1 & (k = j) \\
& & k &= 2 & (k = i) \\
k &= 3 & L(1,3) &= 1 < L(1,2) + L(2,3) = 1 + 1 = 2(\text{true}) & \Rightarrow t = 0 & \Rightarrow s = 0 \\
k &= 4 & L(1,4) &= 1 < L(1,2) + L(2,4) = 1 + 1 = 2(\text{true}) & \Rightarrow t = 0 & \Rightarrow s = 0 \\
k &= 5 & L(1,5) &= 2 < L(1,2) + L(2,5) = 1 + 1 = 2(\text{false}) & \Rightarrow t = \frac{1*1}{1} = 1 & \Rightarrow s = s + t = 1 \\
k &= 6 & L(1,6) &= 2 < L(1,2) + L(2,6) = 1 + 2 = 3(\text{true}) & \Rightarrow t = 0 & \Rightarrow s = 1 \\
j &= 2 & j &= i \\
j &= 3 & k &= 1 & L(3,1) &= 1 < L(3,2) + L(2,1) = 1 + 1 = 2(\text{true}) & \Rightarrow t = 0 & \Rightarrow s = 1 \\
& & k &= 2 & k &= i \\
& & k &= 3 & k &= j \\
k &= 4 & L(3,4) &= 1 < L(3,2) + L(2,4) = 1 + 1 = 2(\text{true}) & \Rightarrow t = 0 & \Rightarrow s = 1 \\
k &= 5 & L(3,5) &= 2 < L(3,2) + L(2,5) = 1 + 1 = 2(\text{false}) & \Rightarrow t = \frac{1*1}{2} = \frac{1}{2} & \Rightarrow s = 1 + \frac{1}{2} = \frac{3}{2} \\
k &= 6 & L(3,6) &= 1 < L(3,2) + L(2,6) = 1 + 2 = 3(\text{true}) & \Rightarrow t = 0 & \Rightarrow s = \frac{3}{2} \\
j &= 4 & k &= 1 & L(4,1) &= 1 < L(4,2) + L(2,1) = 1 + 1 = 2(\text{true}) & \Rightarrow t = 0 & \Rightarrow s = \frac{3}{2} \\
& & k &= 2 & k &= i \\
k &= 3 & L(4,3) &= 1 < L(4,2) + L(2,3) = 1 + 1 = 2(\text{true}) & \Rightarrow t = \frac{1*1}{2} = \frac{1}{2} & \Rightarrow s = \frac{3}{2} + \frac{1}{2} = 2 \\
& & k &= 4 & k &= j \\
k &= 5 & L(4,5) &= 2 < L(4,2) + L(2,5) = 1 + 1 = 2(\text{false}) & \Rightarrow t = 0 & \Rightarrow s = \frac{3}{2} \\
k &= 6 & L(4,6) &= 1 < L(4,2) + L(2,6) = 1 + 2 = 3(\text{true}) & \Rightarrow t = 0 & \Rightarrow s = 2 \\
j &= 5 & k &= 1 & L(5,1) &= 2 < L(5,2) + L(2,1) = 1 + 1 = 2(\text{false}) & \Rightarrow t = \frac{1*1}{1} = 1 & \Rightarrow s = 2 + 1 = 3 \\
& & k &= 2 & k &= i \\
k &= 3 & L(5,3) &= 2 < L(5,2) + L(2,3) = 1 + 1 = 2(\text{false}) & \Rightarrow t = \frac{1*1}{2} = \frac{1}{2} & \Rightarrow s = 3 + \frac{1}{2} = \frac{7}{2} \\
k &= 4 & L(5,4) &= 2 < L(5,2) + L(2,4) = 1 + 1 = 2(\text{false}) & \Rightarrow t = \frac{1*1}{2} = \frac{1}{2} & \Rightarrow s = \frac{7}{2} + \frac{1}{2} = 4
\end{aligned}$$

$$\begin{array}{ll}
k = 5 & k = j \\
k = 6 & L(5,6) = 1 < L(5,2) + L(2,6) = 1 + 2 = 3(\text{true}) \Rightarrow t = 0 \Rightarrow s = 4 \\
j = 6 & k = 1 \quad L(6,1) = 2 < L(6,2) + L(2,1) = 2 + 1 = 3(\text{true}) \Rightarrow t = 0 \Rightarrow s = 4 \\
& k = 2 & k = i \\
k = 3 & L(6,3) = 1 < L(6,2) + L(2,3) = 2 + 1 = 3(\text{true}) \Rightarrow t = 0 \Rightarrow s = 4 \\
k = 4 & L(6,4) = 1 < L(6,2) + L(2,4) = 2 + 1 = 3(\text{true}) \Rightarrow t = 0 \Rightarrow s = 4 \\
k = 5 & L(6,5) = 1 < L(6,2) + L(2,5) = 2 + 1 = 3(\text{true}) \Rightarrow t = 0 \Rightarrow s = 4 \\
k = 6 & k = j
\end{array}$$

$$C_B(2) = \frac{4}{4*5} = 0.2$$

$$\begin{array}{ll}
i = 3 & s = 0 \\
j = 1 & k = 1 \quad k = j \\
& k = 2 \quad L(1,2) = 1 < L(1,3) + L(3,2) = 1 + 1 = 2(\text{true}) \Rightarrow t = 0 \Rightarrow s = 0 \\
& k = 3 & k = i \\
& k = 4 \quad L(1,4) = 1 < L(1,3) + L(3,4) = 1 + 1 = 2(\text{true}) \Rightarrow t = 0 \Rightarrow s = 0 \\
& k = 5 \quad L(1,5) = 2 < L(1,3) + L(3,5) = 1 + 2 = 3(\text{true}) \Rightarrow t = 0 \Rightarrow s = 0 \\
& k = 6 \quad L(1,6) = 2 < L(1,3) + L(3,6) = 1 + 1 = 2(\text{true}) \Rightarrow t = \frac{1*1}{2} = \frac{1}{2} \quad s = 0 + \frac{1}{2} = \frac{1}{2} \\
j = 2 & k = 1 \quad L(2,1) = 1 < L(2,3) + L(3,1) = 1 + 1 = 2(\text{true}) \Rightarrow t = 0 \Rightarrow s = \frac{1}{2} \\
& k = 2 & k = j \\
& k = 3 & k = i \\
& k = 4 \quad L(2,4) = 1 < L(2,3) + L(3,4) = 1 + 1 = 2(\text{true}) \Rightarrow t = 0 \quad s = \frac{1}{2} \\
& k = 5 \quad L(2,5) = 1 < L(2,3) + L(3,5) = 1 + 2 = 3(\text{true}) \Rightarrow t = 0 \Rightarrow s = \frac{1}{2} \\
& k = 6 \quad L(2,6) = 2 < L(2,3) + L(3,6) = 1 + 1 = 2(\text{false}) \Rightarrow t = \frac{1*1}{3} \Rightarrow s = \frac{1}{2} + \frac{1}{3} = \frac{5}{6} \\
j = 3 & j = i \\
j = 4 & k = 1 \quad L(4,1) = 1 < L(4,3) + L(3,1) = 1 + 1 = 2(\text{true}) \Rightarrow t = 0 \Rightarrow s = \frac{5}{6} \\
& k = 2 \quad L(4,2) = 1 < L(4,3) + L(3,2) = 1 + 1 = 2(\text{true}) \Rightarrow t = 0 \Rightarrow s = \frac{5}{6} \\
& k = 3 & k = i \\
& k = 4 & k = j \\
& k = 5 \quad L(4,5) = 2 < L(4,3) + L(3,5) = 1 + 2 = 3(\text{true}) \Rightarrow t = 0 \Rightarrow s = \frac{5}{6} \\
& k = 6 \quad L(4,6) = 1 < L(4,3) + L(3,6) = 1 + 1 = 2(\text{true}) \Rightarrow t = 0 \Rightarrow s = \frac{5}{6}
\end{array}$$

$$\begin{array}{llll}
j = 5 & k = 1 & L(5,1) = 2 < L(5,3) + L(3,1) = 2 + 1 = 3(\text{true}) & \Rightarrow t = 0 \Rightarrow s = \frac{5}{6} \\
& k = 2 & L(5,2) = 1 < L(5,3) + L(3,2) = 2 + 1 = 3(\text{true}) & \Rightarrow t = 0 \Rightarrow s = \frac{5}{6} \\
& k = 3 & & k = i \\
& k = 4 & L(5,4) = 2 < L(5,3) + L(3,4) = 2 + 1 = 3(\text{true}) & \Rightarrow t = 0 \Rightarrow s = \frac{5}{6} \\
& k = 5 & & k = j \\
& k = 6 & L(5,6) = 1 < L(5,3) + L(3,6) = 2 + 1 = 3(\text{true}) & \Rightarrow t = 0 \Rightarrow s = \frac{5}{6} \\
j = 6 & k = 1 & L(6,1) = 2 < L(6,3) + L(3,1) = 1 + 1 = 2(\text{false}) & \Rightarrow t = \frac{1*1}{2} = \frac{1}{2} \Rightarrow s = \frac{5}{6} + \frac{1}{2} = \frac{4}{3} \\
& k = 2 & L(6,2) = 2 < L(6,3) + L(3,2) = 1 + 1 = 2(\text{false}) & \Rightarrow t = \frac{1*1}{3} = \frac{1}{3} \Rightarrow s = \frac{4}{3} + \frac{1}{3} = \frac{5}{3} \\
& k = 3 & & k = i \\
& k = 4 & L(6,4) = 1 < L(6,3) + L(3,4) = 1 + 1 = 2(\text{true}) & \Rightarrow t = 0 \Rightarrow s = \frac{5}{3} \\
& k = 5 & L(6,5) = 1 < L(6,3) + L(3,5) = 1 + 2 = 3(\text{true}) & \Rightarrow t = 0 \Rightarrow s = \frac{5}{3} \\
& k = 6 & & k = j
\end{array}$$

$$C_B(3) = \frac{5/3}{5*4} = 0.083333$$

$$C_R(2) = 0.2 * 0.1875 = 0.0375$$

$$C_R(3) = 0.083333 * 0.214286 = 0.017857$$

We compute BC, C_B , and C_R for the other nodes in the same way, and we

reach to the following result:

$$\begin{array}{lll}
C_B(1) = 0.0 & BC(1) = 0.444444 & C_R(1) = 0.0 \\
C_B(4) = 0.083333 & BC(4) = 0.214286 & C_R(4) = 0.017857 \\
C_B(5) = 0.033333 & BC(5) = 0.857143 & C_R(5) = 0.028571 \\
C_B(6) = 0.10 & BC(6) = 0.333333 & C_R(6) = 0.033333
\end{array}$$

Example(3.2)[13]

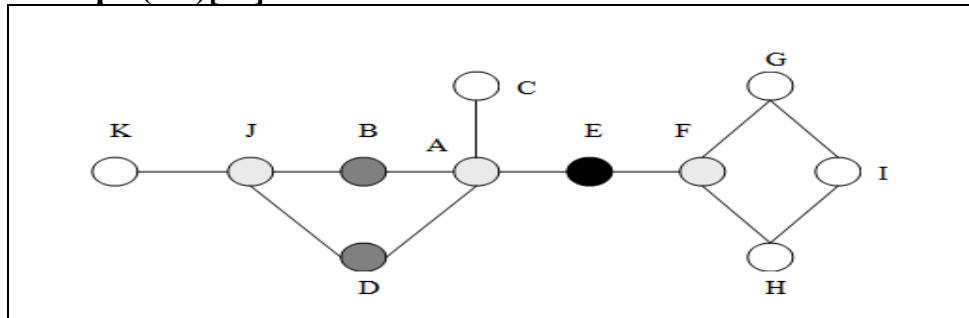


Figure (3.2): Random Unweighted Graph

We apply our Algorithms: Algorithm (3.1) to compute BC and Algorithm (3.3) to compute C_R on the graph in Figure(3.1)

The six nodes that have the highest values of Bridging centrality are:

49

Table(3.1): Values of C_R , BC , C_B For the Graph in Figure (3.1).

Node v	Deg(v)	$C_B(v)$	BC(v)	$C_R(v)$
E	2	0.53333	0.85714	0.45713
B	2	0.15555	0.85714	0.13333
D	2	0.15555	0.85714	0.13333
F	3	0.47777	0.22222	0.10617
A	4	0.65555	0.10000	0.06555
J	3	0.21111	0.16666	0.03519

We compute $C_B(v)$, BC(v) and $C_R(v)$ by using our algorithm.

From these result we arrange the nodes according to the values of the coefficients in an descending order as follows:

- 1- Bridging Coefficient : E, B, D, F, J, A
- 2- Betweenness Centrality : A, E, F, J, D, B
- 3- Bridging Centrality: E, B, D, F, A, J.

Vertex A has the highest degree, thus it has the highest Betweenness Centrality. From the graph we can see that vertex A lies in the center of the graph, and the number of shortest paths that pass through this node is the highest. Vertices E, D and B have the highest value of Bridging Coefficient , from the definition of bridging node ,node E is bridging node in this graph, A and F are also bridging nodes

From this result and after studying many examples, we see that there will be a relation between bridging node and clustering. Our goal is to explore this kind of relation. This work will appear in the next chapter.

3.4 Special Cases:

In the following cases we use V = number of nodes in the graph.

3.4.1. Complete Graph

We state the values of Bridging Centrality (C_R) Bridging Coefficient (BC) and Betweenness Centrality (C_B) in the case of the complete graph

a- The betweenness centrality $C_B(v) = 0$ for all $v \in V$

Proof:

In complete graph every node is adjacent to all other nodes. This means that length of a shortest path between any two nodes is equal to one.

$$d_G(s,t) = 1 \text{ for all } s, t \in V \text{ } s \neq t]$$

$$d_G(s,v) + d_G(v,t) = 2$$

From equation(3.4) $\sigma_{st}(v) = 0$ for all $s, t \in V$.

Then from equation(3.3) $C_B(v) = 0$ for all $v \in V$.

b- The bridging coefficient is

$$BC(v) = \frac{1}{n-1} \quad (3.5)$$

Proof :

$$BC = \frac{D(v)^{-1}}{\sum_{i \in N(v)} D(i)^{-1}}$$

$$D(v) = n - 1 \quad \forall v \in V$$

$$BC(v) = \frac{(n-1)^{-1}}{\frac{n-1}{n-1}} = \frac{(n-1)^{-1}}{1} = (n-1)^{-1} = \frac{1}{n-1}$$

c- The bridging centrality is $C_R(v) = 0$ for all $v \in V$

Proof :

$$C_R(v) = BC(v) * C_B(v)$$

$$CB(v) = 0 \quad \text{for all } v \in V \text{ in complete graph}$$

$$C_R(v) = 0 \quad \text{for all } v \in V \text{ in complete graph}$$

3.4.2. Complete Bi-partite Graph

We state the values of Bridging Centrality (C_R) Bridging Coefficient (BC) and Betweenness Centrality (C_B) in the case of the complete bi-partite graph.

Let n_1 = order of V_1 and n_2 = order of V_2 . Then we have the following:

$$\text{a- } C_B(v) = \frac{(1/n_i) * (n_j - 1) * (n_j)}{(n-1)(n-2)} \quad v \in V_i \quad (3.6)$$

Proof:

Let $v \in V_1$ then:

$$\sigma_{st} = \begin{cases} n_2 & s, t \in V_1 \\ n_1 & s, t \in V_2 \\ 1 & s \in V_1, t \in V_2 \end{cases}$$

$$\begin{aligned}
\text{And } \sigma_{st}(v) &= \begin{cases} 0 & s, t \in V_1 \\ 1 & s, t \in V_2 \\ 0 & s \in V_1, t \in V_2 \end{cases} \\
\Rightarrow \frac{\sigma_{st}(v)}{\sigma_{st}} &= \begin{cases} 0 & s, t \in V_1 \\ 1 & s, t \in V_2 \\ n_1 & s \in V_1 \text{ and } t \in V_2 \\ 0 & s \in V_1 \text{ and } t \in V_2 \end{cases} \\
\Rightarrow C_B(v) &= \frac{\sum_{\substack{s, t \in V_2 \\ s \neq t \neq v}} \left(\frac{\sigma_{st}(v)}{\sigma_{st}} \right)}{((n-1)(n-1))}
\end{aligned}$$

There are $(n_2 * (n_2 - 1))$ pairs of $s, t \in V_2$, then:

$$C_B(v) = \frac{\left(\frac{1}{n_1} \right) * (n_2) * (n_2 - 1)}{(n-1) * (n-2)}$$

In the same way we prove C_B for node $v \in V_2$ as follow:

$$\begin{aligned}
C_B(v) &= \frac{\left(\frac{1}{n_2} \right) * (n_1) * (n_1 - 1)}{(n-1) * (n-2)} \\
\Rightarrow C_B(v) &= \frac{\left(\frac{1}{n_i} \right) * (n_j) * (n_j - 1)}{(n-1) * (n-2)} \quad v \in V_i
\end{aligned}$$

b- Bridging Coefficient :

$$BC(v) = \frac{n_i}{n_j^2} \quad v \in V_i \quad (3.7)$$

Proof :

Let $v \in V_1$

$$BC = \frac{D(v)^{-1}}{\sum_{i \in N(v)} D(i)^{-1}}$$

$$D(v) = n_2$$

$$N(v) = \{i, i \in V_2\}$$

$$D(i) = n_1$$

$$\Rightarrow BC(v) = \frac{\frac{1}{n_1} * n_2}{\frac{1}{n_2}} = \frac{n_1}{n_2}$$

c- Bridging Centrality :

$$C_R(v) = BC(v) * C_B(v)$$

$$C_R(v) = \frac{(n_j - 1)}{(n - 1)(n - 2)(n_j)} \quad v \in V_1 \quad (3.8)$$

3.4.3. Star graph

We state the values of Bridging Centrality (C_R) Bridging Coefficient (BC) and Betweenness Centrality(C_B) in the case of the star graph.

We consider the star as a special case of the bipartite

$$n_1 = 1, n_2 = (n - 1)$$

For the center:

a- Betweenness Centrality

$$C_B(v) = \frac{(1/n_1) * (n_2 - 1) * (n_2)}{(n - 1)(n - 2)}$$

$$C_B(v) = \frac{(1/1) * (n - 2) * (n - 1)}{(n - 1)(n - 2)}$$

$$C_B(v) = 1$$

b- Bridging Coefficient $BC(v) = \frac{n_1}{n_2}$

$$BC(v) = \frac{1}{(n - 1)^2} \quad (3.9)$$

c- Bridging Centrality:

$$C_R(v) = \frac{1}{(n - 1)^2} \quad (3.10)$$

For the other nodes

a- Betweenness Centrality

$$C_B(v) = \frac{(1/n_1) * (n_2 - 1) * (n_2)}{(n-1)(n-2)}$$

$$C_B(v) = \frac{(1/(n-1)) * (1-1) * (1)}{(n-1)(n-2)} \Rightarrow C_B(v) = 0$$

b- Bridging Coefficient

$$BC(V) = \frac{n_2}{n_1^2} = \frac{(n-1)}{1} \Rightarrow BC(v) = (n-1) \quad (3.11)$$

c- Bridging Centrality :

$$C_R(v) = 0$$

3.4.4. Cycle graph

We state the values of Bridging Centrality (C_R) Bridging Coefficient (BC) and Betweenness Centrality (C_B) in the case of the cycle graph

a- Betweenness Centrality

$$C_B(v) = \begin{cases} \frac{(\frac{n}{2}-1) + 2 \sum_{i=2}^{\frac{n}{2}-1} (\frac{n}{2}-i)}{(n-1)(n-2)}, & n \text{ is even} \\ \frac{2 \sum_{i=1}^{\frac{n-3}{2}} (\frac{n-1}{2}-i)}{(n-1)(n-2)}, & n \text{ is odd} \end{cases} \quad (3.12)$$

We reach to this result by several iteration with different weight of graphs.

Example (3.3):

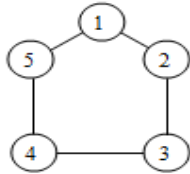


Figure (3.3): 5-Cycle Graph.

The adjacency matrix for the graph is:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The number of shortest path between any pair of nodes given in the array (Num) and the length of shortest path between any pair of nodes given in array (L).

$$Num = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} 0 & 1 & 2 & 2 & 1 \\ 1 & 0 & 1 & 2 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 1 & 0 & 1 \\ 1 & 2 & 2 & 1 & 0 \end{bmatrix}$$

Compute C_B for node (1):

The maximum shortest path in Figure (3.3) = 2 (the maximum shortest path = $\frac{(n-1)}{2}$), each pair of nodes have only one shortest path between, the

shortest paths that pass through node (1) are: the path from node 2 to node

5, and the path from node 5 to node 2, there are two shortest paths pass through node 1 (or $\frac{(n-1)}{2}$ shortest path pass from node 1). Then:

$$\begin{aligned} \sum_{\substack{s \neq t \neq 1 \\ s, t, 1 \in V}} \frac{\sigma_{st}(v)}{\sigma_{st}} &= \frac{\sigma_{2,3}(1)}{\sigma_{2,3}} + \frac{\sigma_{2,4}(1)}{\sigma_{2,4}} + \dots + \frac{\sigma_{5,4}(1)}{\sigma_{5,4}} \\ &= \frac{0}{1} + \frac{0}{1} + \frac{1}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{1}{1} + \frac{0}{1} + \frac{0}{1} = 2 \end{aligned}$$

$$C_B(1) = \frac{2}{(5-1)(5-2)} = 0.166667$$

In the same way we compute C_B for the other nodes in the graph and we find that they have the same value.

Example (3.4):

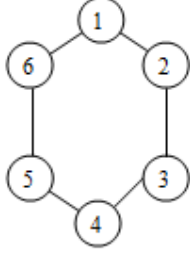


Figure (3.4): 6-Cycle Graph.

The adjacency matrix for the graph is:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The number of shortest path between any pair of nodes given in the array (Num) and the length of shortest path between any pair of nodes given in array (L).

$$Num = \begin{bmatrix} 0 & 1 & 1 & 2 & 1 & 1 \\ 1 & 0 & 1 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 2 & 1 & 1 & 0 & 1 & 1 \\ 1 & 2 & 1 & 1 & 0 & 1 \\ 1 & 1 & 2 & 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 1 \\ 1 & 0 & 1 & 2 & 3 & 2 \\ 2 & 1 & 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 & 1 & 2 \\ 2 & 3 & 2 & 1 & 0 & 1 \\ 1 & 2 & 3 & 2 & 1 & 0 \end{bmatrix}$$

Compute C_B for node (1):

The maximum shortest path in Figure (3.3) = 3 (the maximum shortest path = $\frac{n}{2}$), each pair of nodes have either one or two shortest path between,

the shortest paths that pass through node (1) are the path between each pair of the following: (2,5),(5,2),(2,6),(6,2),(3,6), and (6,3) there are six shortest paths pass through node 1(or n shortest path pass from node1), four pairs have two shortest path one pass through node 1.

Then:

$$\sum_{\substack{s \neq t \neq 1 \\ s, t, 1 \in V}} \frac{\sigma_{st}(v)}{\sigma_{st}} = \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{1} + \frac{1}{1} = 4$$

$$C_B(v) = \frac{4}{(6-1)(6-2)} = 0.20$$

In the same way we compute C_B for the other nodes in the graph and we find that they have the same value.

We apply the same operation for cycle graph gave order (7,8,9,10) until we reach to the result in equation (3.12).

b- Bridging Coefficient

$$BC(v) = 0.5$$

Proof :

$$BC = \frac{D(v)^{-1}}{\sum_{i \in N(v)} D(i)^{-1}}$$

$D(v) = 2$ for any vertex v in the Cycle.

$$BC(v) = \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{2}} = 0.5$$

c- Bridging Centrality:

$$C_R(v) = BC(v) * C_B(v)$$

$$C_R(v) = 0.5 * \begin{cases} \frac{(\frac{n}{2}-1) + 2 \sum_{i=2}^{\frac{n}{2}-1} (\frac{n}{2}-i)}{(n-1)(n-2)}, & n \text{ is even} \\ \frac{2 \sum_{i=1}^{\frac{n-3}{2}} (\frac{n-1}{2}-i)}{(n-1)(n-2)}, & n \text{ is odd} \end{cases}$$

3.4.5. Path graph

We state the values of Bridging Centrality (C_R) Bridging Coefficient (BC) and Betweenness Centrality (C_B) in the case of the path graph

a. Betweenness Centrality :

$$C_B(v_i) = \frac{2(n-i)(i-1)}{(n-1)(n-2)} \quad (3.13)$$

where v denotes the number of nodes

Proof :

In the path any node v is in the shortest path between any two nodes that lie on the different side of v , and there is only one shortest path between any two nodes. Node v divides the path into two parts:

Part one contains $(i-1)$ nodes, while part two contains $(n-i)$ nodes.

Each node in part one has only one shortest path to any node in part two,

and the opposite is true, and this path (shortest path) passes through node v .

$$C_B(v) = \sum_{\substack{s \neq t \neq v \\ s, t, v \in V}} \frac{\sigma_{st}(v)}{\sigma_{st}} \bigg/ ((n-1) * (n-2))$$

$$\frac{\sigma_{st}(v)}{\sigma_{st}} = 1 \quad \forall \quad s, t \in V \text{ and from different part}$$

$$\Rightarrow C_B(v_i) = \frac{2(n-i)(i-1)}{(n-1)(n-2)}$$

b- Bridging Coefficient

$$BC(v) = \begin{cases} 2 & v = 1, v = n \\ 0.333333 & v = 2, v = (n-1) \\ 0.5 & otherwise \end{cases}$$

Proof :

In path $D(1) = 1$, $D(n) = 1$, $D(v) = 2$ for all other nodes.

$$BC = \frac{D(v)^{-1}}{\sum_{i \in N(v)} D(i)^{-1}}$$

$$BC(1) = \frac{1/1}{1/2} = 2 \quad ,$$

$$BC(2) = \frac{1/2}{\frac{1}{1} + \frac{1}{2}} = \frac{1}{3} = 0.333333$$

$$BC(v) = \frac{1/2}{\frac{1}{2} + \frac{1}{2}} = 0.5$$

Chapter Four

Clustering Analysis in Unweighted Graphs

4.1 Unweighted Graph

An unweighted graph is a graph whose edges have no values. When we deal with unweighted graph we consider the weight of each edge to be equal to one. When we want to apply the method of cluster to this type of graph, the clustering methods that we saw in chapter two are not suitable because they depend on the weight of the graph. There are other methods of clustering that depend on the properties of the graph not on the weight of the graph, such as the connectivity of the graph as in HCS clustering algorithm.

4.2. Highly Connected Subgraph Clustering Algorithm

Highly Connected Subgraph (HCS) algorithm of clustering depends on the connectivity of the graph.

Connectivity of the graph G , denoted by $k(G)$, is the minimum number of edges whose removal gives a disconnected graph. The set of removed edges is called a cut. If $k(G)=L$, then the graph G is called an L -connected graph.

HCS algorithm identifies highly connected Subgraph as cluster (We consider the graph highly connected if $k(G) > \frac{N}{2}$, but single vertices are

not considered clusters and they are grouped in a singleton set

(S)[11][19] .

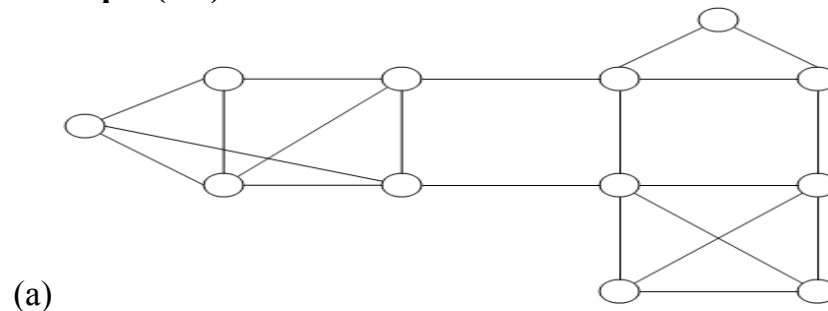
Algorithm (4.1): HCS Clustering Algorithm[19]

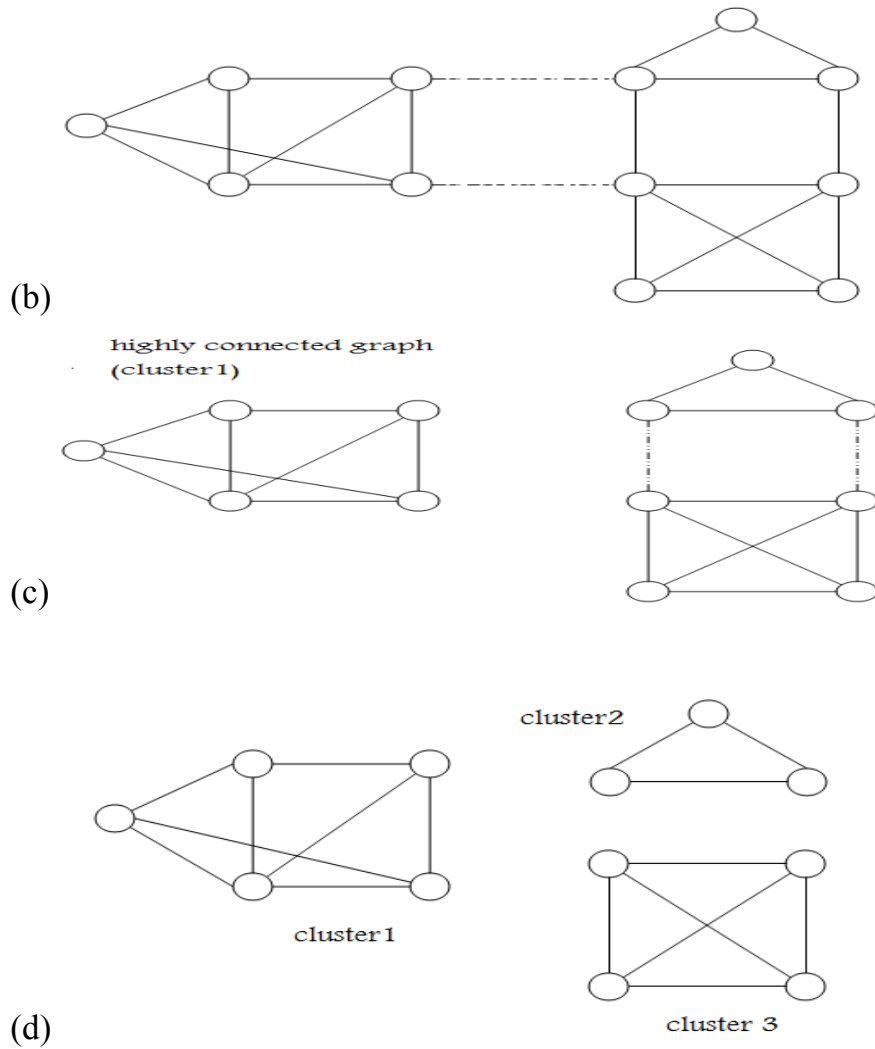
```

HCS(G(V,E))
begin
  (H, H', C) ← MINCUT(G)
  if G is highly connected
    then return (G)
  else
    HCS(H)
    HCS(H')
  end if
end

```

The running time of HCS algorithm is $2X \cdot f(N, E)$ where X denotes the number of clusters and $f(N, E)$ denotes the time complexity of computing a minimum cut in a graph with N vertices and E edge) ^[19].

Example (4.1)



Figure(4.1): An Example on HCS Clustering Algorithm

We apply HCS clustering algorithm on the graph on Figure(4.1)

(a) the graph.

(b) minimum cut edges are denoted by broken line.

(c) after first cut the first Subgraph is highly connected

($k(H) = 3, \frac{5}{2} = 2.5, \quad 3 > 2.5$) this is the first cluster.

(d) after the other minimum cut there are 3 subgraph each subgraph is highly connected then there are 3 cluster.

4.2.1 Properties of HCS Clustering [19]:

In this respect, we note the following observations:

1- The diameter of every highly connected graph is at most two.

The diameter of connected graph G , $\text{diam}(G)$, is the maximum distance between any two vertices in the graph G .

2- Any two vertices are either adjacent or share one or more common neighbors.

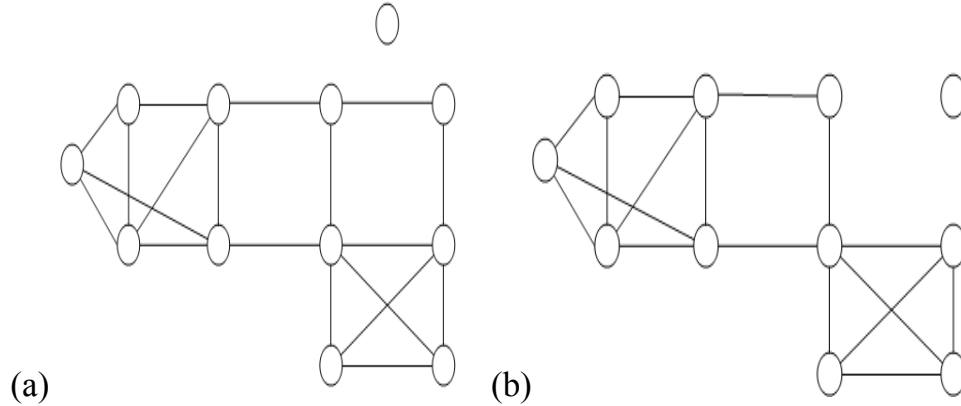
3- It shows a strong indication of homogeneity.

4- Any non-trivial set split by the algorithm has diameter at least three.

4.2.2 Modified HCS Algorithm

When there are several minimum cuts in the graph, the algorithm might choose a minimum cut which is not best from a clustering point view. In many cases this process will break the cluster into singletons [11].

Consider Graph (4.1) again. If we choose another cut, and consider vertices with minimum degree, the result will be as the following:



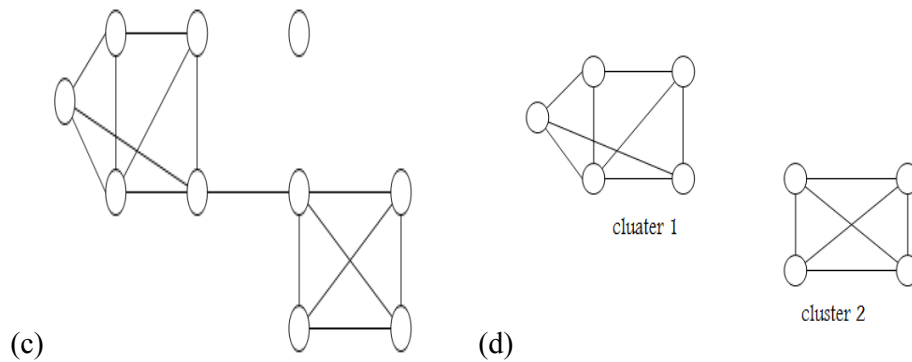


Figure (4.2): Applying HCS Clustering Algorithm To the Graph in Figure (4.1).

We can see that the three nodes which form cluster 2 in the previous example are taken as singletons, in HCS clustering algorithm there is no rule to choose the best minimum cut. To solve this problem we modify HCS algorithm as the follows:

Perform several iteration of the HCS algorithm until no new cluster is found [19] .

4.3 HCS Analysis

After we study HCS clustering algorithm, and apply it to many algorithms we reach to the following result.

First: The HCS clustering algorithm depends on the connectivity of the graph, inn the following graph $k(G) = 1$

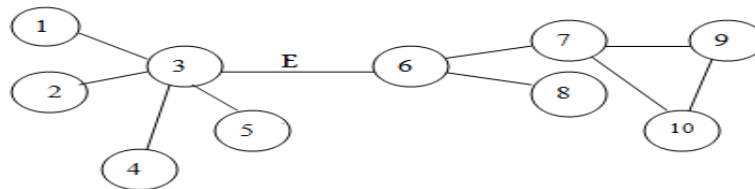


Figure (4.3) Random Unweighted Graph

If we apply HCS clustering algorithm, most of the nodes in the graph will be taken as singletons.

Second: It is clear that the cut will be for the edge E. If we cut this edge, the result will be 2 clusters, $C1(1,2,3,4,5)$, $C2(6,7,8,9,10)$. But when we look at C1 we find that $k(C1) = 1$ and $N(C1) = 5$ ($N(C1)$ = number of nodes in C1) which gives $k(C1) < \frac{N}{2}$. This means that C1 is not highly connected and we can't consider it as a cluster in HCS algorithm and the same thing is true for C2.

So I see that can modify HCS algorithm as follows:

- 1- The edge that we chose to cut must be between two nodes whose degrees are greater than one.
- 2- Give more properties for the cluster.

4.4 Properties of Cluster

In all method of clusters the main point was how to decide the number of clusters, and when we can consider the subgraph a cluster.

After deep studying to different methods of clustering we find that the cluster must have one of these properties:

Suppose P is a subgraph, and N is the order of P :

- 1- Connectivity of graph ($k(P) > \frac{N}{2}$).
- 2- The maximum shortest path between any two nodes must be at most 2 ($diam(P) \leq 2$) .

3- When the average clustering coefficient of the nodes in the graph equal zero and there is a node adjacent to at least 40% of the nodes and the other nodes have degree at most 2 and $diam(P)$ is not greater than 4 ($diam(P) \leq 4$) in this case we consider the subgraph a cluster .

Clustering coefficient (C): is the number of triangle around the node divide by number of expected triangle

$$C = \frac{t}{\sum_{i=1}^{d-1} i} \quad (3.1)$$

Where d is the degree of node, t number of triangle around the node.

(in Figure(4.3) the walk (7, 9, 10, 7) is a triangle around node 7, the expected triangles around node 7 are (7,9,10,7), (7,9,8,7), (7,9,6,7), (7,10,8,7), (7,10,6,7), (7,8,6,7). $ClusteringCoefficient(7) = \frac{1}{6}$).

4.5 Analysis

In the previous chapter we discussed the bridging nodes and we saw that the bridging node lies between modules and it has the highest bridging centrality.

Scale-free network is a graph that has a small number of nodes with high degrees and these nodes are adjacent to nearly 70% of other nodes. The other nodes have small degrees. When we apply any clustering method to this kind of graph we find that each of the nodes that have high degree lies in a cluster and the bridging nodes in this kind of graph is very clear. Consider the following graph.

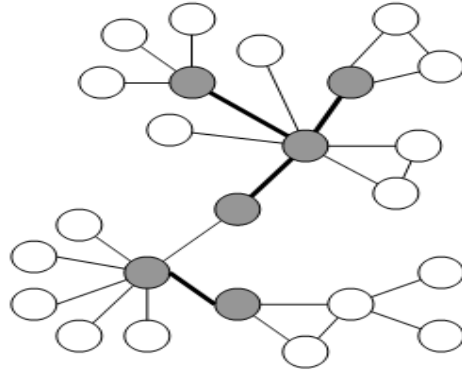


Figure (4.4): Scale-Free Network.

Figure (4.4) presents a scale-free network where the gray nodes have the highest bridging centrality. If we cut the edges between these nodes, referring bold edges, we will have an isolated graph and each part can be considered a cluster.

From that result we find that we can reach to isolated modules if we cut the edges between the nodes that have high bridging centrality and these modules can be considered clusters when we deal with scale-free network.

4.6 Highest Bridging Centrality Cut Algorithm

After deep studying of clustering method and several iterations, we reach to a new algorithm for clustering that depends on bridging centrality, C_R , of the nodes in the graph. This algorithm computes the bridging centrality for each node and then finds the highest 25% of values ($x = \max(C_R) - (\max(C_R) - \min(C_R))/4$). Then for each node that has C_R greater than x , it finds the highest C_R for its neighbor and cuts the

edges between these two nodes. But there is the main condition for cut: we can't cut an edge between two nodes one of them having $C_R = 0$.

The module P is defined as a cluster if it satisfies one of the following properties :

- 1- ($diam(P) \leq 2$) .
- 2 - Average clustering coefficient= 0 and there is a node adjacent to at least 40% of the nodes and the other nodes have degree at most two. In this case we consider the subgraph as a cluster.

We called this algorithm Highest Bridging Centrality Cut algorithm ($HC_R C$ Algorithm).

Algorithm (4.2): The $HC_R C$ Clustering Algorithm:

```

 $HC_R C(G)$ 
   $C_R \rightarrow$  array contain bridging centrality /* $C_R(i)$ =bridging centrality
                                                    for node  $i$ */

   $x = \max(C_R(i)) - (\max(C_R(i)) - \min(C_R(i)) / 4)$ 
  for  $i = 1:N$  //N number of nodes in the graph
    if  $C_R(i) \geq x$ 
      cut the edge between node  $i$  and  $\max(C_R(\text{neighbor of } i))$ 
    end
  end

   $t$  = number of parts that the graph split into
  for  $i = 1:t$ 
    if  $P(i)$  is a cluster //the properties of the cluster that we discuss

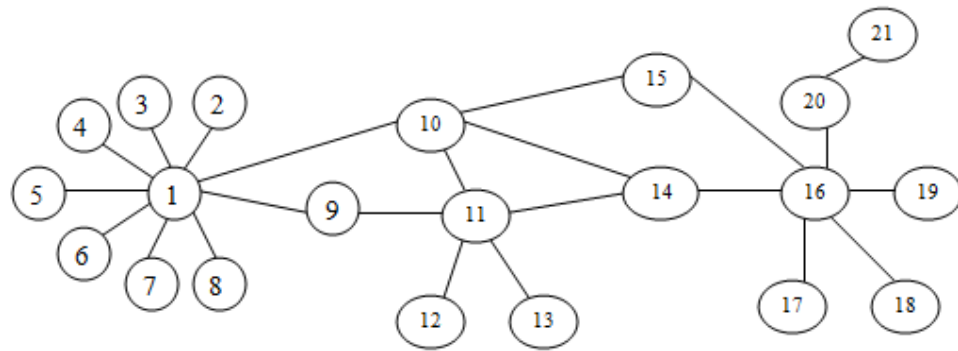
```

```

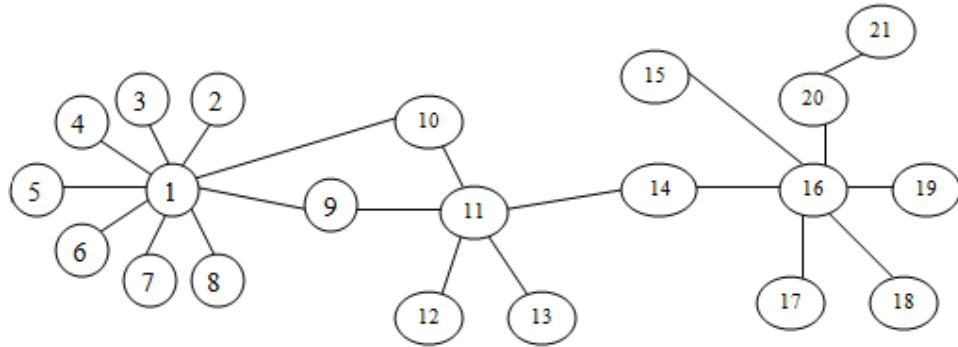
P(i) is a cluster
Else
  HCRC (P(i))
end
end
end

```

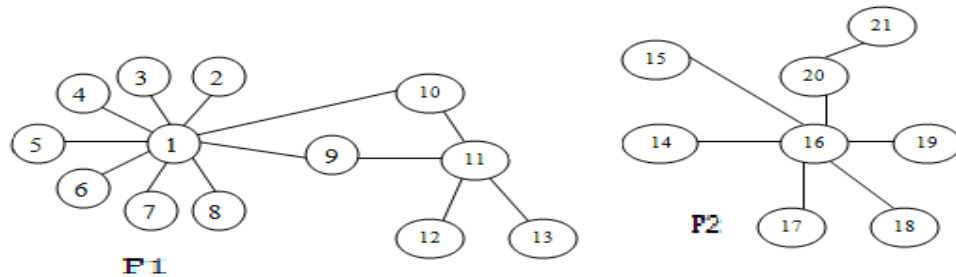
Example (4.2):



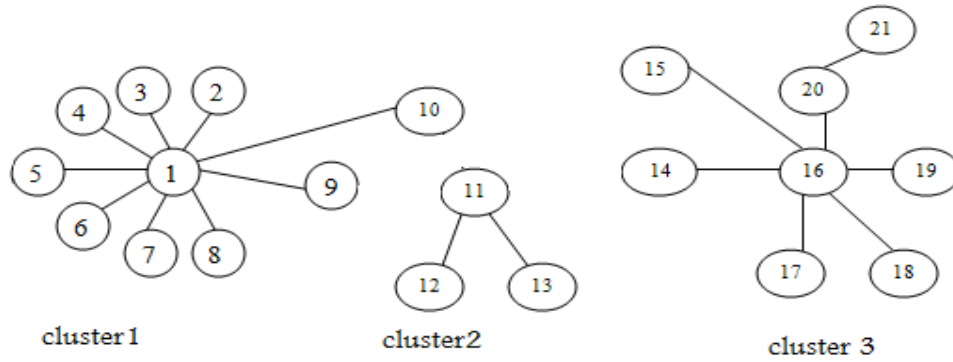
A



B



C



D

Figure (4.5): Applying HC_RC Algorithm for Clustering.

Applying HC_RC algorithm:

(A) Step 1:

Compute C_R for each node:

Node	1	2	3	4	5	6	7
C_R	0.008527	0.0	0.0	0.0	0.0	0.0	0.0
Node	8	9	10	11	12	13	14
C_R	0.000000	0.101504	0.092552	0.015363	0.0	0.0	0.145092
Node	15	16	17	18	19	20	21
C_R	0.170526	0.017105	0.0	0.0	0.0	0.042857	0.0

$$x = 0.170526 - \left(\frac{0.170526 - 0.0}{4} \right) = 0.127895$$

There are two nodes having values of C_R greater than 0.127895. These nodes are 15 and 14.

$Ne(15) = \{10, 16\}$, $\max(C_R) = 0.092552$ and this is for node 10.

Then we cut the node between nodes 15 and 10.

$Ne(14) = \{10, 11, 16\}$, $\max(C_R) = 0.092552$ and this is for node 10.

Then we cut the node between node 14 and 10.

The symbol $Ne(i)$ denotes the neighbors of node i , i.e the set of nodes that are adjacent to node i .

After cutting these two edges the graph still consist of one part.

(B) Step 2

compute C_R for the graph

Node	1	2	3	4	5	6	7
C_R	0.008224	0.0	0.0	0.0	0.0	0.0	0.0
Node	8	9	10	11	12	13	14
C_R	0.0	0.37218	0.37218	0.035338	0.0	0.0	0.65311
Node	15	16	17	18	19	20	21
C_R	0.0	0.017193	0.0	0.0	0.0	0.042857	0.0

$$x = 0.65311 - \frac{(0.65311 - 0.0)}{4} = 0.489433$$

$Ne(14) = \{11, 16\}$, $\max(C_R) = 0.035338$ and this is for node 11. Then we cut the edge between nodes 14, 11.

After step 2, we find that the graph splits into 2 parts, P1 and P2. we note that P1 is not cluster(it doesn't satisfy any of clusters properties that we apply) while P2 is a cluster and it satisfies the third property.

(C) Step 3:

Compute C_R for P1 .

Node	1	2	3	4	5	6	7
C_R	0.01189	0.0	0.0	0.0	0.0	0.0	0.0
Node	8	9	10	11	12	13	

C_R	0.0	0.251748	0.251748	0.027146	0.0	0.0	
-------	-----	----------	----------	----------	-----	-----	--

$$x = 0.251748 - \frac{(0.251748 - 0.0)}{4} = 0.188811$$

$Ne(9) = \{1, 11\}$, $\max(C_R) = 0.027146$ and this value is for node 11.

The we cut the edge between nodes 9 and 11.

$Ne(10) = \{1, 11\}$, $\max(C_R) = 0.027146$ and this is for node 11. The we cut the edge between nodes 10 and 11.

We see that P_1 splits into 2 parts P_{11} and P_{12} where each of them is a cluster and it satisfies the second property.

Here we stop.

4.7 Properties of $HC_R C$ Algorithm

After making the comparison between $HC_R C$ algorithm and several different algorithms we find that this algorithm has the following properties:

- 1- The average running time of this algorithm is smaller than other algorithms.
- 2- The cluster is homogeneous if it satisfies properties one or two.
- 3- Property 3 gives the algorithm more validity exactly when we deal with scale-free network. This is because in scale-free network, when we apply clustering method some modules of the graph will have one center and the other nodes will have degree at most two. This is well seen in the following graph.

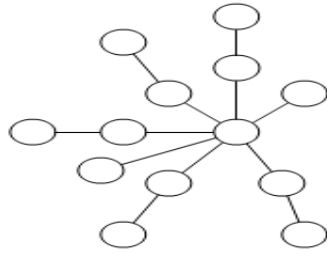


Figure (4.6): Small Unweighted Undirected Graph

If we want to split this graph by applying any of the clustering methods, the result will be very small parts, consisting of two nodes.

So the third condition gives this algorithm more validity for properties of clusters.

4.8 Comparison Between HCS Algorithm and HC_RC Algorithm

After applying HCS algorithm and HC_RC algorithm to many different networks, we find that there are several differences between these two algorithms.

1- Applying to scale-free network

We can't apply HCS algorithm to scale-free network because of the following reasons:

- a- The minimum cut in scale-free network is equal to one. So applying this algorithm will result in many singletons.
- b- The number of clusters will be very large and the average size will be very small, because the clusters must be highly connected.

HC_RC algorithm is suitable for scale-free network.

2- Applying to random network

Both HCS algorithm and $HC_R C$ algorithm are suitable for random networks.

3- Running time

$HC_R C$ algorithm running time is smaller than the HCS algorithm running time.

The running time for HCS algorithm = $2X * f(N, E)$ and the modified HCS algorithm will take at least twice this time and depending on the different clusters that occur in each iteration

We can't give main equation to the running time of $HC_R C$ algorithm because it depends on the place of bridging nodes in each iteration.

Example (4.3)

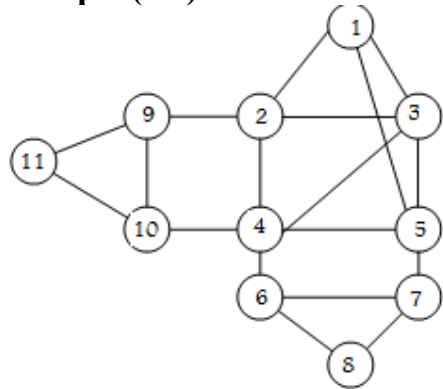


Figure (4.7) random unweighted graph

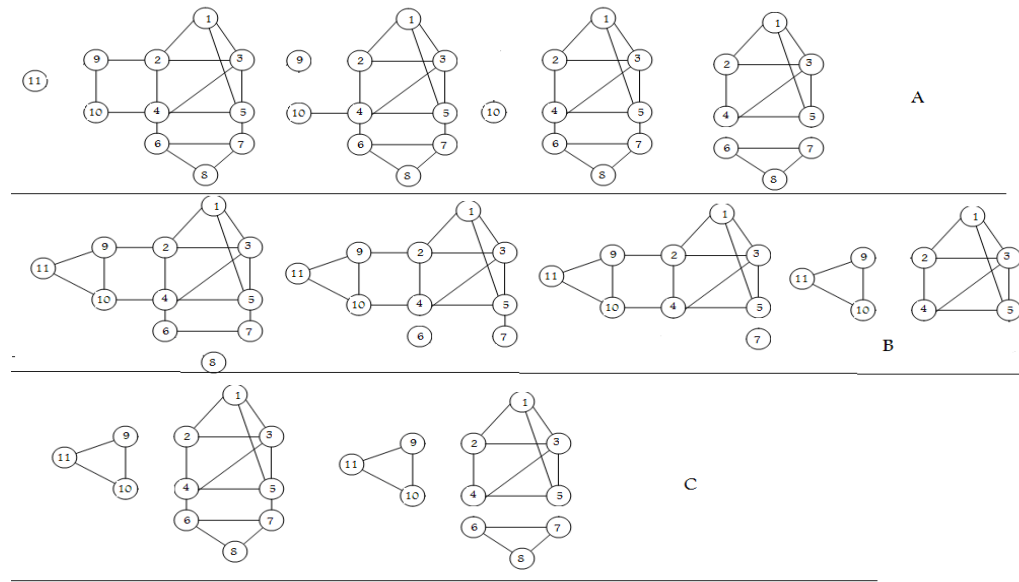


Figure (4.8): Apply HCS Algorithm to Graph in Figure (4.7),

When we apply HCS algorithm to the graph on Figure (4.7) the minimum cut comes to be equal to two. There are many minimum cuts. The first iteration A we get two clusters. The second iteration B we get two clusters, one is new. The third iteration C we get three clusters but there is no new cluster. Then we have three clusters.

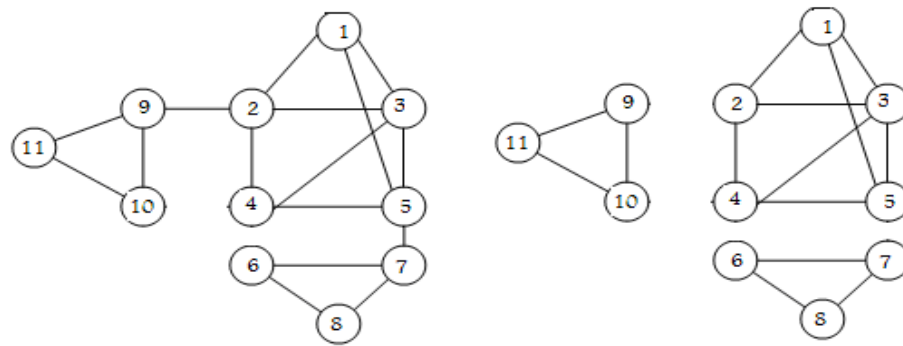


Figure (4.9): Apply HCRC Algorithm to the Graph in Figure (4.7)

When we apply HCRC clustering algorithm to the graph in Figure (4.7) the highest 25% of C_R are the edge between nodes 4 and 10, and the

edge between nodes 4 and 6, in the second iteration the highest 25% of C_R are the edges between nodes 9 and 2, and the edge between nodes 5 and 7. We reach to the same clusters that we reach to when we use HSC clustering algorithm only in two steps.

4.9 Conclusion

The main purpose in this thesis was to find new clustering algorithms depending on the Bridging Centrality of the graph.

There are different clustering algorithms depend on the weight of the graph. We can split these algorithms into two main categories:

Hierarchical clustering algorithm (such as Single linkage method) and Partitioned clustering algorithm (such as K-means algorithm) these methods are suitable for weighted graphs.

But when we want to cluster unweighted graphs we must deal with other algorithms which depend on the properties of the graph not on the weight of the edges. One of these algorithms was Highly Connected Subgraph (HCS) algorithm of clustering, that algorithm depends on the connectivity of the graph. In this algorithm we cut the number of edges whose removal disconnects the graph, and identifies highly connected subgraph as cluster. But single vertices are not considered clusters and they are grouped in a singleton set (S). Some times when there are several minimum cuts in the graph, the algorithm might chose a minimum cut which is not best from a clustering point view. In many

cases this process will break the cluster into singletons. Modified HCS algorithm solve this problem by performing several iteration of the HCS algorithm until no new cluster is found.

But if we apply this algorithm to scale-free network in most times the minimum cut will be equal to one, so we will have many single nodes. To solve this problem we suggest some conditions on the cut and on the properties for the cluster.

After deep studying and iterations we reach to a new algorithm for clustering. This algorithm depends on the Bridging Centrality of the graph. This algorithm cut the edges between the nodes that have highest Bridging Centrality. And we define the subgraph as a cluster if it satisfies one of the conditions that we defined in section (4.4).

Example(4.4)

Large unweighted graph

By applying $HC_R C$ Algorithm on large graph.

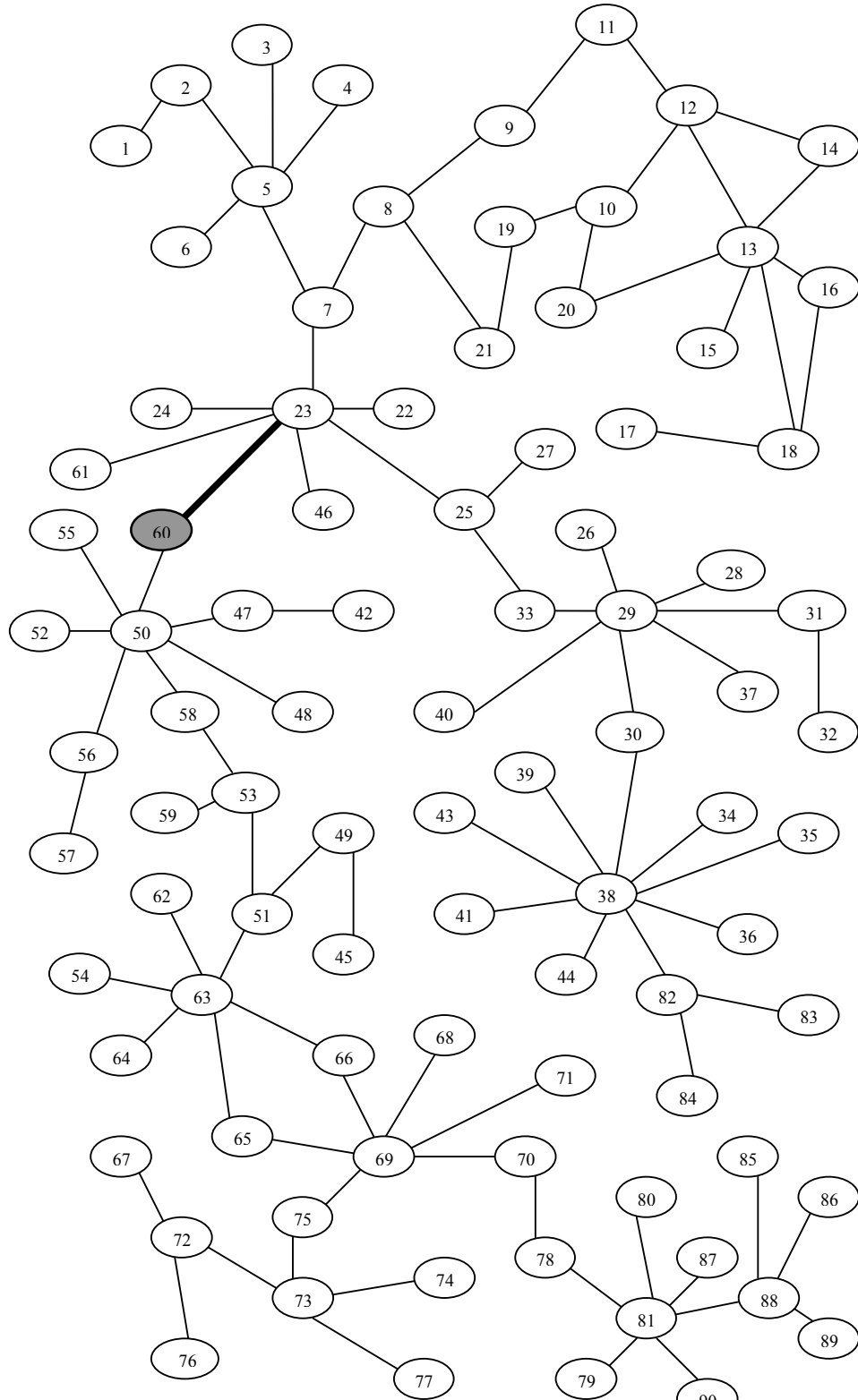


Figure (4.10): Large Unweighted Scale-Free Network Example (4.4) step1.

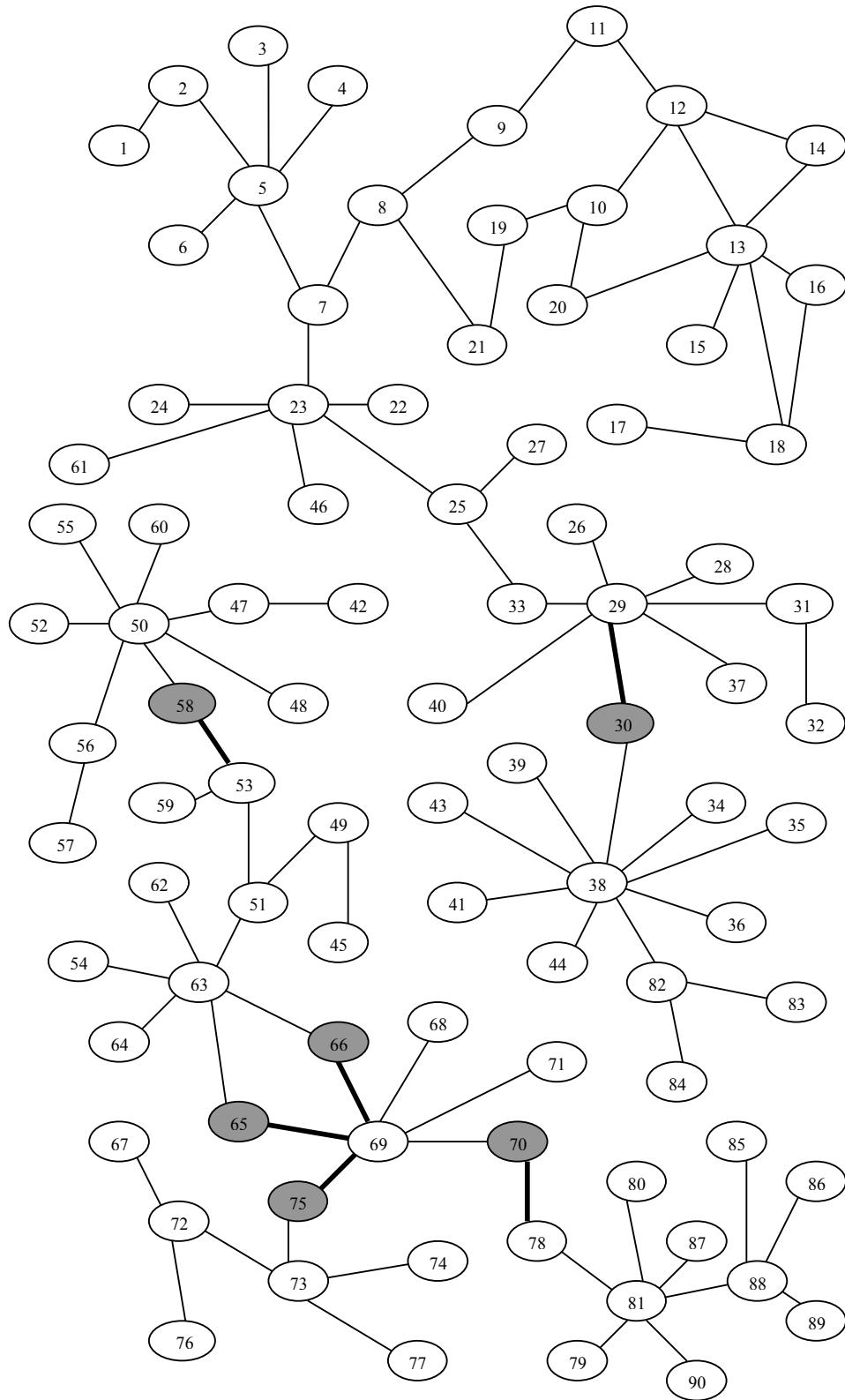


Figure (4.11): Large Unweighted Scale-Free Network Example (4.4) Step2.

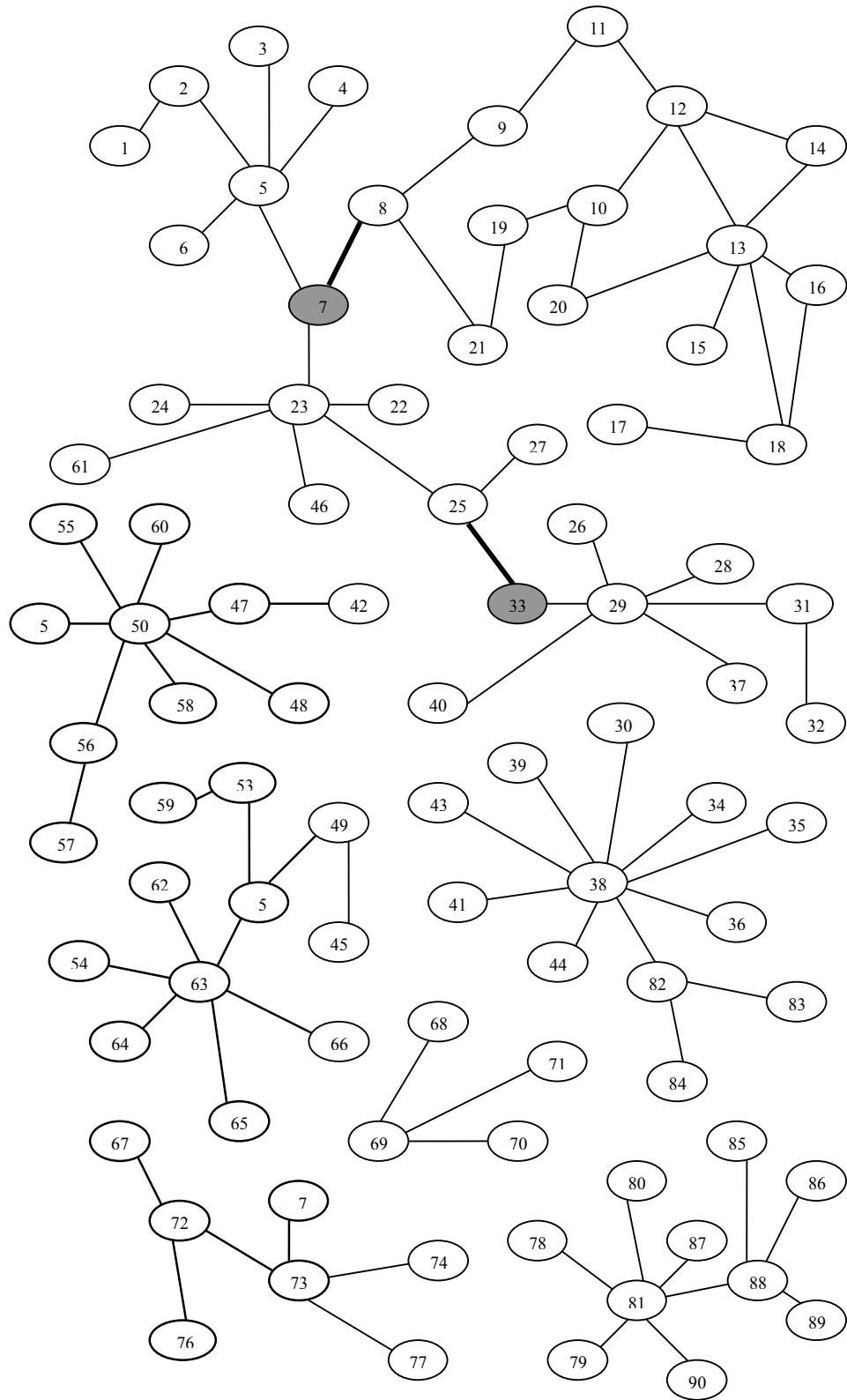


Figure (4.12): Large Unweighted Scale-Free Network Example (4.4) Step3.

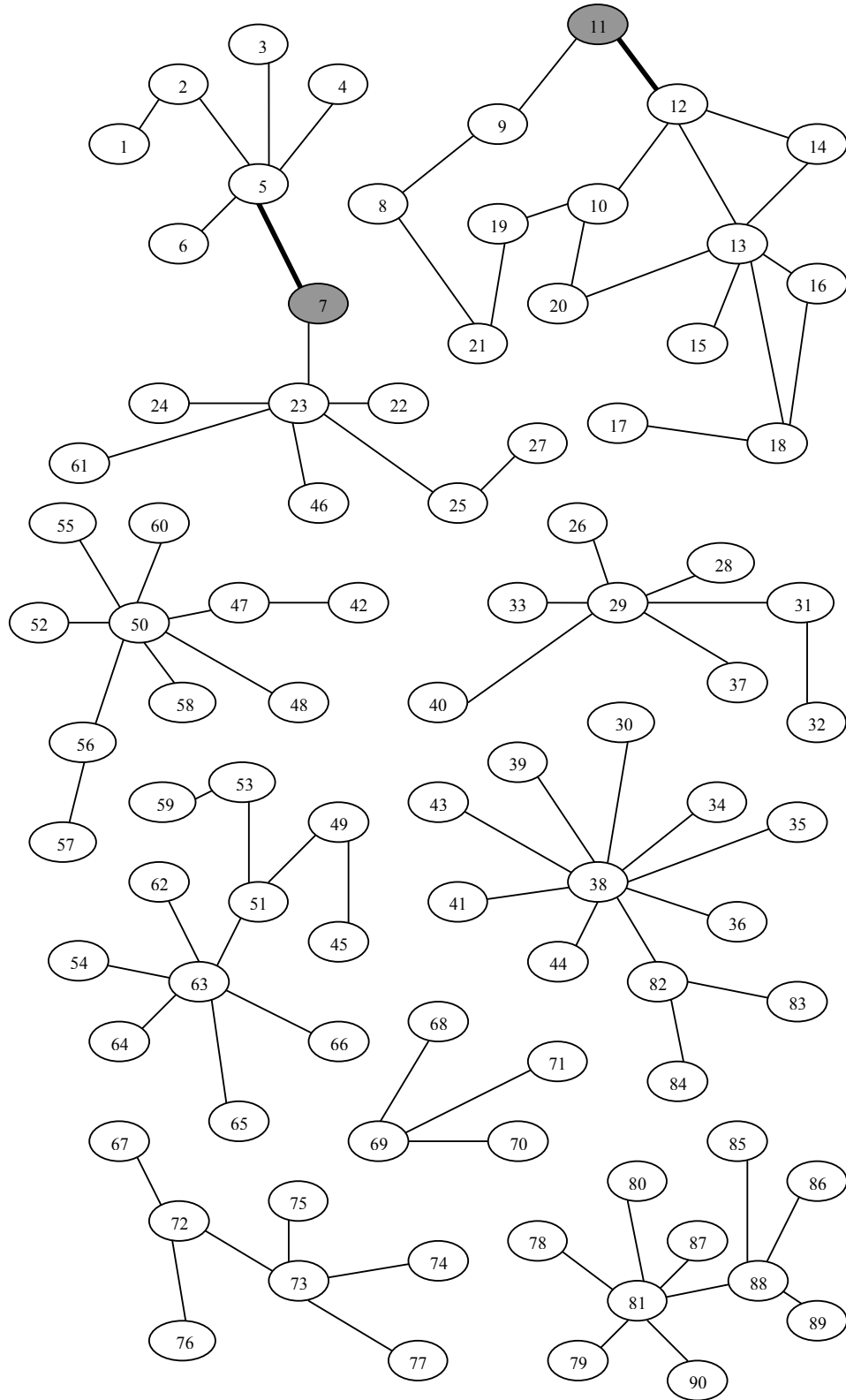


Figure (4.13): Large Unweighted Scale-Free Network Example (4.4) Step 4.

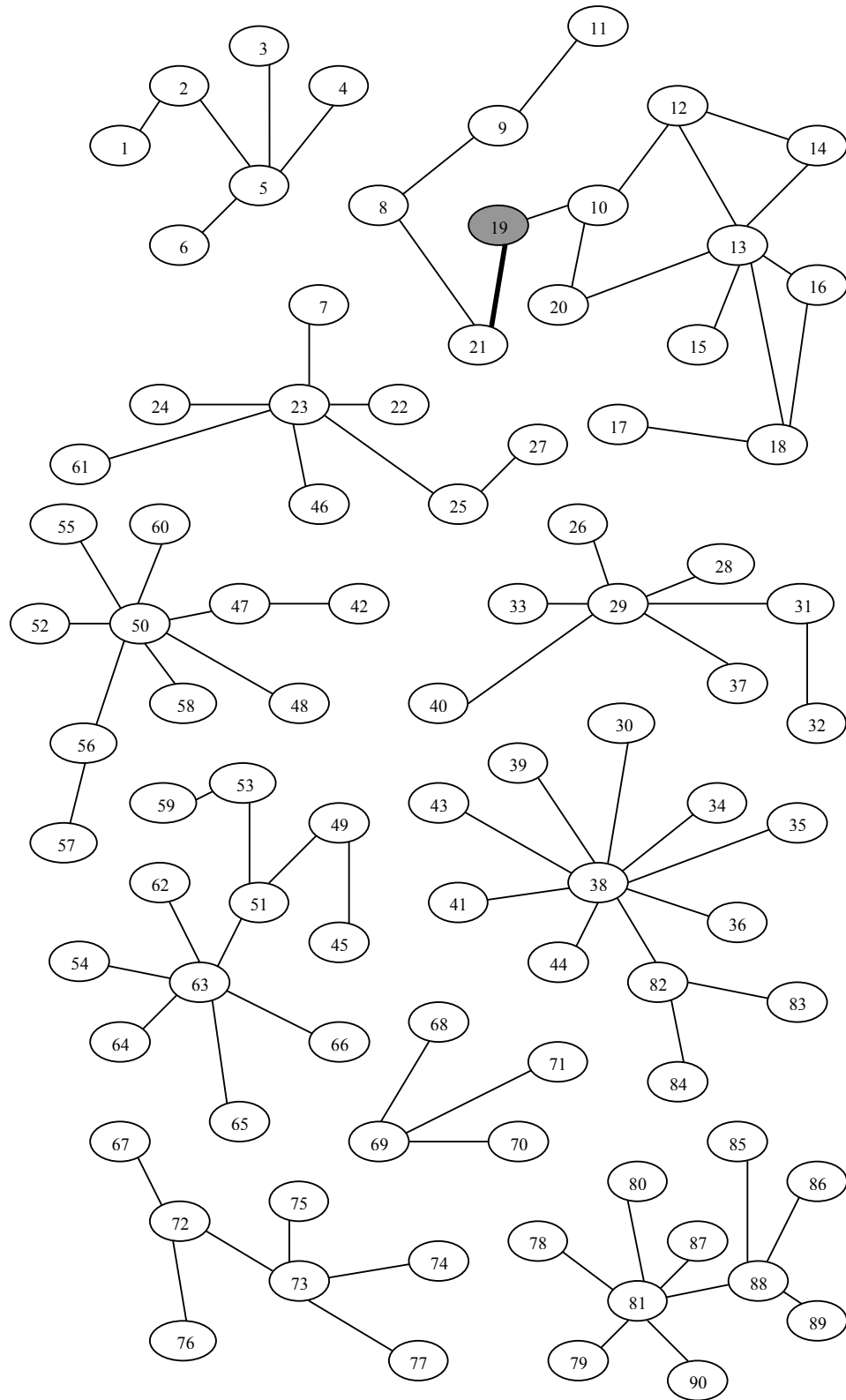


Figure (4.14): Large Unweighted Scale-Free Network Example (4.4) Step 5.

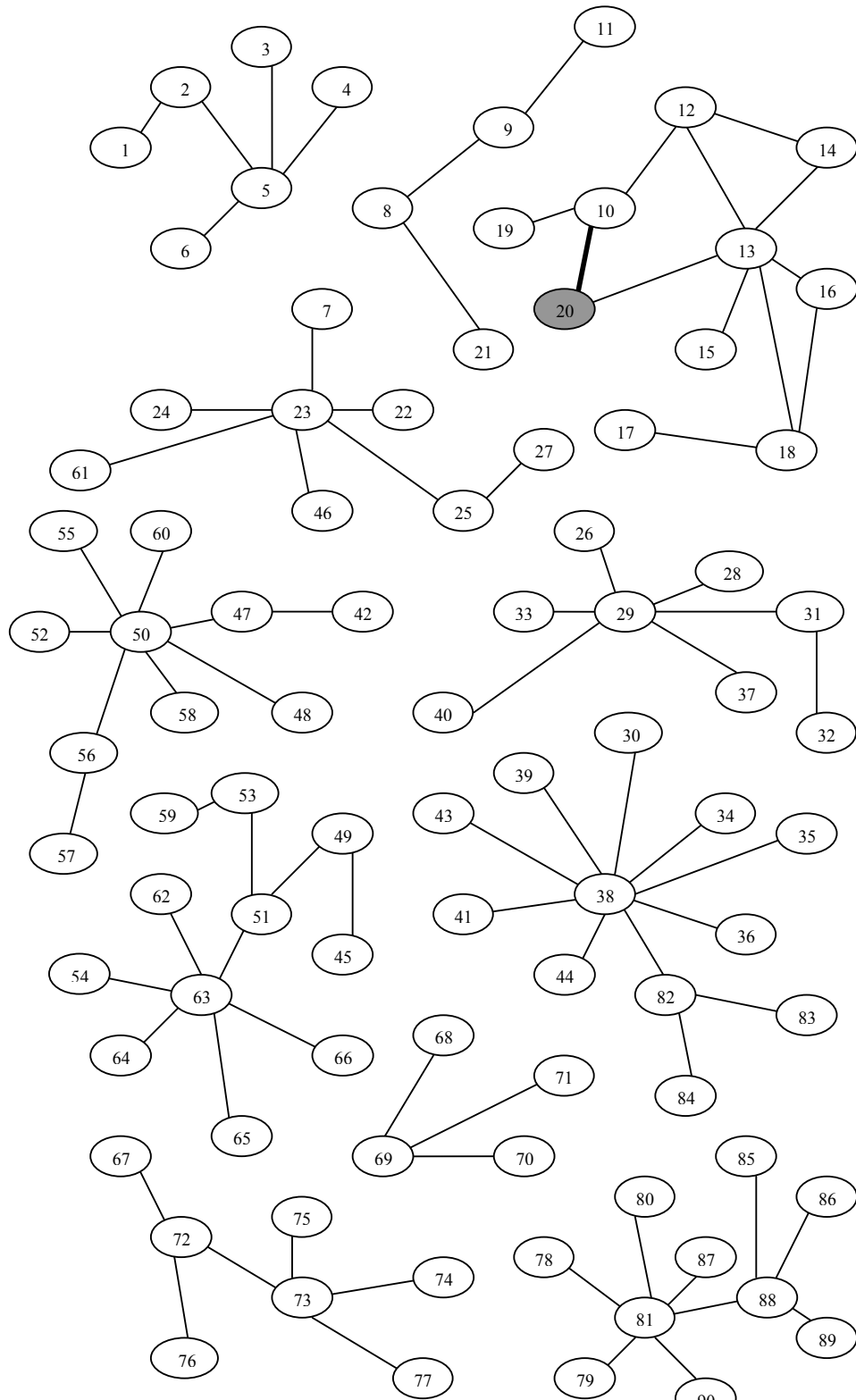


Figure (4.15): Large Unweighted Scale-Free Network Example (4.4) Step 6.

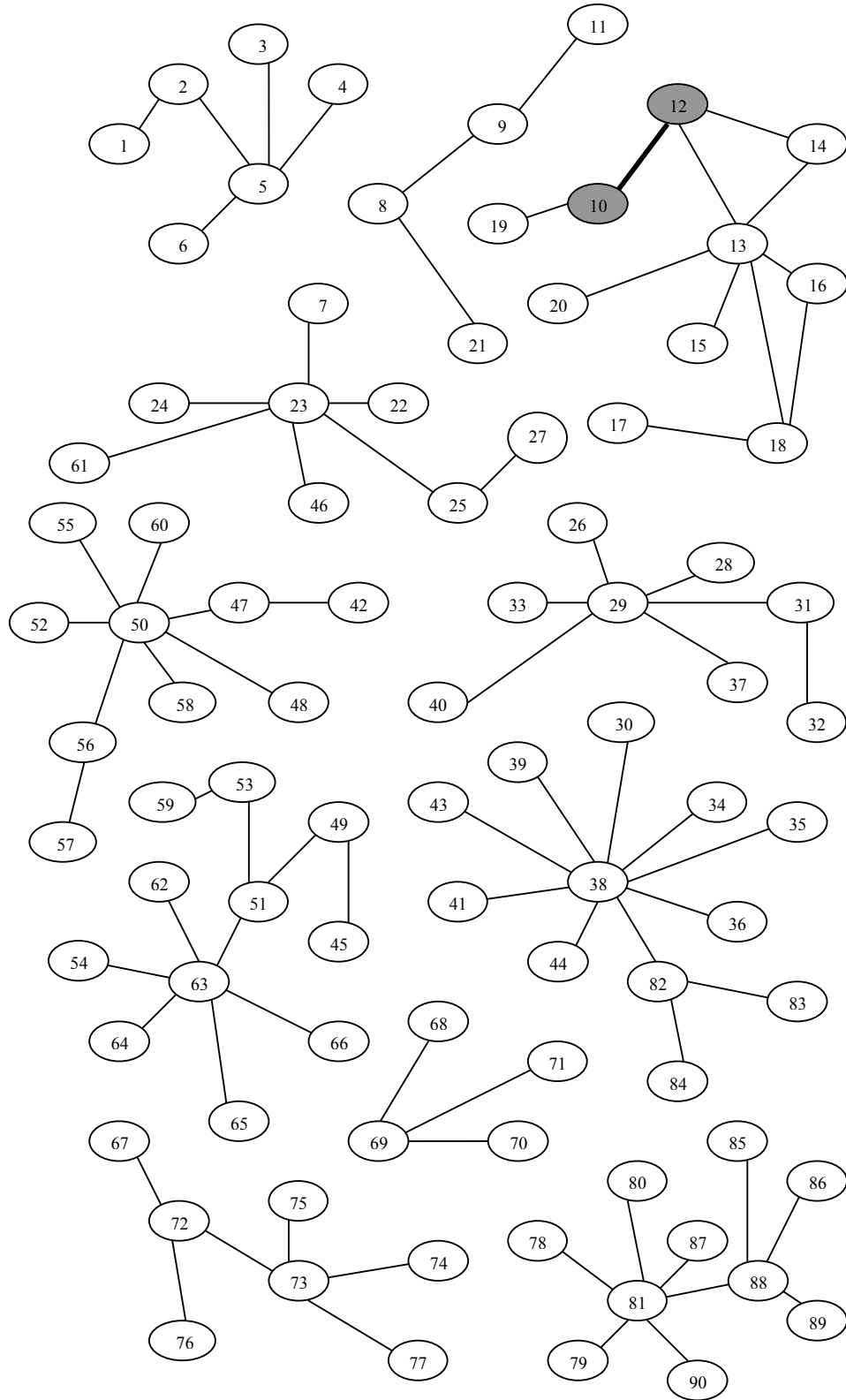


Figure (4.16): Large Unweighted Scale-Free Network Example (4.4) Step 7.

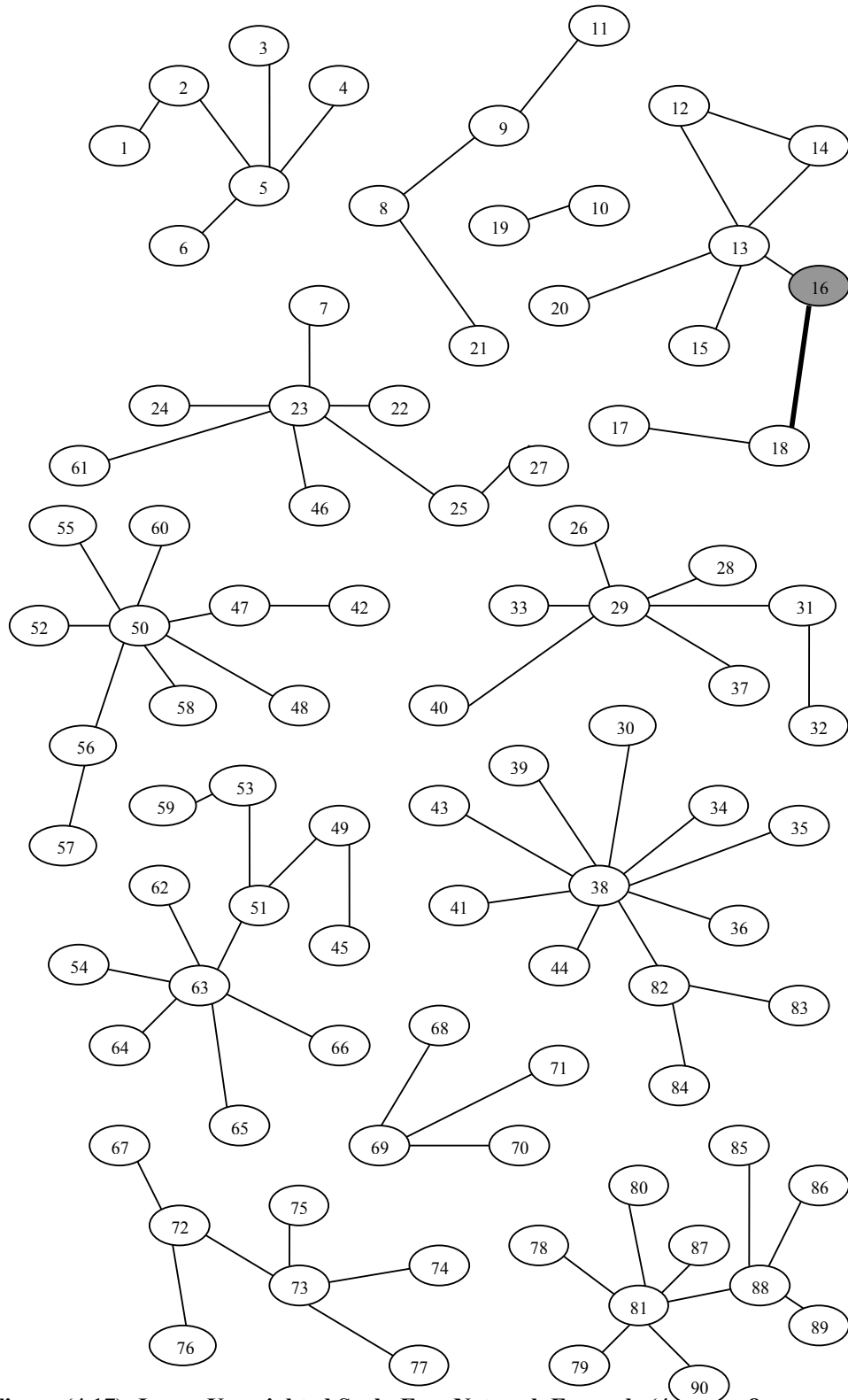


Figure (4.17): Large Unweighted Scale-Free Network Example (4.4) Step8.

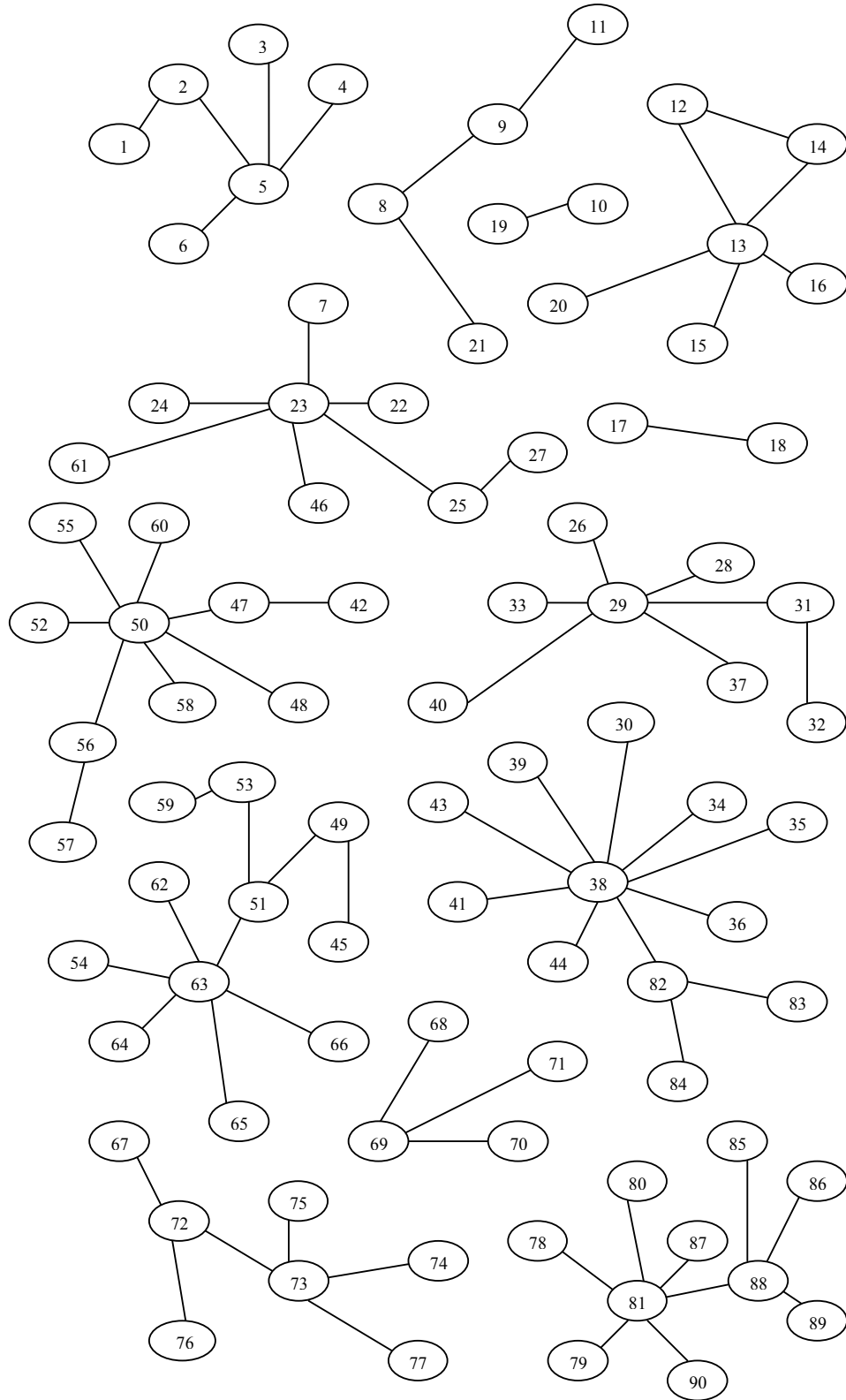


Figure (4.18): Large Unweighted Scale-Free Network Example (4.4) Step 9.

When we apply HC_RC clustering algorithm the clustering will be as following:

Step1: when we calculate C_R for all the nodes in Figure (4.13) we find that the highest 25% of C_R is for node (60) we cut the edge between this node and the highest C_R of its neighbors. The graph split into two parts P_1 and P_2 , but we can't consider one of them a cluster.

Step 2: we calculate C_R to the two parts we reach to in step1, in part one we find that the highest 25% of C_R is for nodes (58,65,66,75,70) we cut the edges between these nodes and the nodes that have highest C_R of their neighbors. And in P_2 we find that the highest 25% of C_R is for node (30).

We repeat this operation until each sub graph consider as a cluster, satisfy the properties in section(4.4)

In this example we reach to 13 clusters.

References

- 1) A Tutorial on Clustering Algorithms ,**K-means clustering**,
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html
- 2) A. K. Jain and R. C. Dubes **Cluster Analysis** (Jan, 2003).
- 3) A.Hidalgo R. and Albert-Laszlo Barabasi .**Scale-free network**.
Scholar pedia, the free peer reviewed encyclopedia (2008);
3(1):1616.
- 4) Andrew .W. Moore **K-means and hierarchical Clustering** (Nov
16th, 2001)
- 5) Andrew Moore: “**K-means and Hierarchical Clustering - Tutorial
Slides**”. <http://www-2.cs.cmu.edu/~awm/tutorials/kmeans.html>
- 6) Barabasi, A.L. Dezso, Z. Ravasz, E. yook, S.H. and Oltvai, Z. **Scale-
free and hierarchical structures in complex network**,(2002).
- 7) Brandes, U. **A fastest Algorithm For Betweenness Centrality**,
(2001).
- 8) Chien.C.C. **Hierarchical Clustering** , (May 2008)
- 9) David.S. and Marten. G. **Clustering of EEG-Segments Using
Hierarchical Agglomerative Methods and Self-Organizing**, (2001)
- 10) Erez .H. and Ron.S. **HCS Clustering Algorithm based on Graph
Connectivity** ,(March 1999).
- 11) Hartuv,M. and Shamir, R. **A clustering Algorithm basrd on
Graph Connectivity** ,(1999).

- 12) Hierarchical Clustering method Example,
<http://ocw.mit.edu/NR/rdonlyres/Health-Sciences-and-Technology/HST-508Genomics-and-Computational-BiologyFall2002/CBC2DFF7-3872-4BCA-B0C0-EC09CDC24102/0/hierarchical.pdf>
- 13)Hwang, W. Cho,Y. Zhang, A. and Ramanathan, M. **Bridging Centrality: Identifying Bridging Nodes In Scale-Free Network.**(2006)
http://www.scholarpedia.org/article/Scale-free_networks
- 14)Jan matlis, **Quick Study : scale-free network** , November 4,2002.
- 15)Kaski. S. **Data Exploration Using Self-Organizing Maps.** Part 2.1 Clustering Methods.(1997).
- 16) S.C.Johnson **Hierarchical Clustering Algorithm** (1967)
- 17)Scott Gasch ,The algorithm Archive, 0.4.3 **Floyd's Algorithm – Shortest Paths.** (1999).
<http://www.fearme.com/misc/alg/node88.html>
- 18)Stefano Mossa, Marc Barthélemy, H. Eugene Stanley, and Luís A. Nunes Amaral, **Truncation of Power Law Behavior in "scale-Free" Network Models due to Information Filtering,** March 4, 2002.
- 19) Thiagarajan, B. **A Clustering Algorithm based on Graph Connectivity,** (2001)
- 20)Wikipedia ,free encyclopedia . **Scale-free network,** (28 May 2009).
http://en.wikipedia.org/wiki/Scale-free_network

- 21) Wikipedia ,free encyclopedia, **K-means Clustering**, (30 June 2009). http://en.wikipedia.org/wiki/K-means_clustering
- 22) Wikipedia ,free encyclopedia, **Clustering Coefficient** ,(25 June 2009). http://en.wikipedia.org/wiki/Clustering_coefficient
- 23)Zhang, A. Ramanathan, M. Hwang, W. and Cho,Y. **Bridging centrality : a concept and formula to identify bridging nodes in scale-free network**. (2007).

Appendices

Appendix (A)

My Matlap Programs:

program to find C_R (Bridging centrality)

```

Clc
clear all
n=input('enter number of nodes in the graph: ');
A=enter(n);
BC=BC(A,n);
CB=CB(A,n);
for i=1:n
    CR(i)=CB(i)*BC(i);
end
fprintf('CB\t\tBC\t\tCR\n');
for i=1:n
    fprintf('%f\t%f\t%f\n',CB(i),BC(i),CR(i));
end

```

Note: $A = \text{enter}(n)$ is a function to enter the adjacency matrices. You can choose the kind of graph (star, tree, bipartite, line, or random graph). $BC(A,n)$ is a function to find Bridging coefficient od

the graph. $C_B(A,n)$ is a function to find Betweenness Centrality of the graph.

Function Enter

```
function [A]=enter(n)
fprintf('1- star \n2- Cycle \n3- Complete graph \n4- Path \n5- Complete
Bi-partite \n6- Random graph\n')
s=input('Enter number of graph kind\n');
if s==1
    A=zeros(n);
    A(1,:)=1;
    A(:,1)=1;
    A(1,1)=0;
else if s==2
    A=zeros(n);
    A(1,2)=1;
    A(1,n)=1;
    A(n,1)=1;
    A(n,n-1)=1;
    for q=2:(n-1)
        A(q,q-1)=1;
        A(q,q+1)=1;
    end
else if s==3
```

```

A=zeros(n);
for k=1:n
    A(k,:)=1;
    A(k,k)=0;
end
else if s==4
    A=zeros(n);
    A(1,2)=1;
    A(n,n-1)=1;
    for r=2:(n-1)
        A(r,r-1)=1;
        A(r,r+1)=1;
    end
else if s==5
    x1=input('number of nodes in part one');
    x2=n-x1;
    A=[zeros(x1),ones(x1,x2);ones(x2,x1),zeros(x2)];

else if s==6
    for t=1:n
        for l=t:n
            fprintf('entry %d %d:',t,l)
            A(t,l)=input("");
            A(l,t)=A(t,l);

```

```

                                End
                            end
                        end
                    end
                End
            end
        end
    end
End

```

Function to find Bridging Coefficient

```
function [BC]=BC(A,n)
```

```
for i=1:n
```

```
    N(i)=0;
```

```
    for j=1:n
```

```
        if A(i,j)==1;
```

```
            N(i)=N(i)+1;
```

```
        end
```

```
    end
```

```
end
```

```
%BC
```

```
for i=1:n
```

```
    t=0;
```

```
    for j=1:n
```

```

    if A(i,j)==1
        t=t+1/N(j);
    end
end
BC(i)=(1/N(i))/t;
End

```

Function to find Betweenness Centrality

```

function [CB]=CB(A,n)
Num=A;%number of shortest path
L=A;%length of shortest path
for k=2:n-1
    B=A^k;
    for i=1:n
        for j=1:n
            if Num(i,j)==0
                Num(i,j)=B(i,j);
                L(i,j)=k;
                Num(j,i)=Num(i,j);
                L(j,i)=L(i,j);
            end
        end
    end
    if i==j
        Num(i,j)=0;
        L(i,j)=0;
    end
end

```

```

        End
    end
end
end
%calculate Bridging Centrality ( $C_B$ )
for i=1:n
    s=0;
    for j=1:n
        if j~=i
            for k=1:n
                if k~=i
                    if  $L(j,k) < L(j,i) + L(i,k)$ 
                        t=0;
                    else
                         $t = \text{Num}(j,i) * \text{Num}(i,k) / \text{Num}(j,k);$ 
                    end
                    s=s+t;
                end
            end
        end
    end
     $C_B(i,1) = s / ((n-1) * (n-2));$ 
End

```


Floyd's Algorithm

```
Clc
```

```
clear all
```

```
n=input('Enter number of vertices\n');
```

```
for i=1:n
```

```
    for j=1:n
```

```
        fprintf('entry %d %d',i,j)
```

```
        A(i,j)=input("");
```

```
    end
```

```
end
```

```
for i=1:n
```

```
    for j=1:n
```

```
        if A(i,j)==0 && i~=j
```

```
            I(i,j,1)=Inf;
```

```
        else
```

```

        I(i,j,1)=A(i,j);

    end

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for k=1:n

    for i=1:n

        for j=1:n

            if I(i,j,k)<(I(i,k,k)+I(k,j,k))

                I(i,j,k+1)=I(i,j,k);

            else

                I(i,j,k+1)=I(i,k,k)+I(k,j,k);

            end

        end

    end

end

I

```

Appendix (B)

The Adjacency Matrix For Small Unweighted Graph in Figure(3.1)

[illegible]

Appendix (C)

[illegible]

[illegible]

Column 31-60:

[illegible]

[illegible]

[illegible]

[illegible]

Column 61-90:

[illegible]

[illegible]

[illegible]

[illegible]

ب

التجسير المركزي في الشبكات الحرة
باستخدام عقدة الجسر كمرجع للتجميع

إعداد

هند علي احمد عيد

إشراف

د. صبحي رزية

الملخص

نظرية الرسوم هو احد اشهر المواضيع في الرياضيات, وذلك لأهمية تطبيقاته في حل الكثير من المشاكل, كما ان استخدام هذه التطبيقات يساعدنا في فهم ودراسة بعض الظواهر الطبيعية.

يركز هذا العمل بشكل اساسي على دراسة الشبكات الحرة وخصائصها. بالإضافة الى انه ركز على دراسة طرق التجميع مع محاولة التوصل الى خوارزمية جديدة للتجميع مستخدما خصائص الشبكات الحرة. وكذلك التجسير المركزي لكل عقدة في الشبكة.

ان التجسير المركزي يعتمد على البنية المركزية وكذلك معامل التجسير, وبعد دراسة عميقة لهذه القيم في الشبكات الحرة لاحظنا وجود علاقة بين هذه القيم لعقدة معينة وموقعها في الشبكة.

في النهاية تمكنا من ايجاد العلاقة بين قيمة التجسير المركزي لكل عقدة وموقعها في الشبكة, وتمكنا من ايجاد خوارزمية جديدة للتجميع تعتمد على التجسير المركزي لكل عقدة.

جامعة النجاح الوطنية

كلية الدراسات العليا

التجسير المركزي في الشبكات الحرة
باستخدام عقدة الجسر كمرجع للتجميع

إعداد

هند علي احمد عيد

إشراف

د. صبحي رزية

قدمت هذه الأطروحة استكمالاً لمتطلبات الحصول على درجة الماجستير في
الرياضيات المحوسبة بكلية الدراسات العليا في جامعة النجاح الوطنية في نابلس،
فلسطين.

2010م