Network Data Visualization and Distributed Average Algorithm

Kewei Zhang 1064198

December 15, 2020

Chapter 1: Introduction

In this chapter, the basic ideas of graph will be introduced. Graph is one prime mathematical object, the research on graph can have many different applications. The definition of graph, with the two basic type of graph which are directed graph and undirected graph are introduced in this section, some simple graphs will be applied as example to help understand mentioned notions. Then the essential matrices method to present graph are introduced.

1. Graph Theory

A graph of mathematics is a series of points, discrete or continuous, as in forming a curve or surface, each of which represents a value of a given function. There are two types of graph which are directed graph and undirected graph.

Direction

The difference between an undirected graph and a directed graph is the edge type. An edge has a direction or flow are called as directed edge, respectively the edge has no direction is undirected edge. If all the edges in a graph are directed, the graph is said to be a directed graph, also called digraph. If all the edges in a graph are undirected, the graph is said to be an undirected graph. Fig 1.0 is one example of undirected graph, on the right part of Fig 1.0, we have the mathematical expression of the graph. And Fig. 2 is one example of directed graph, the mathematical expression of the graph is on the right part.

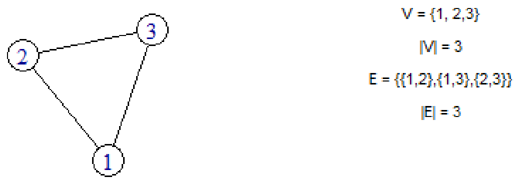

Figure 1: Undirected simple graph Example

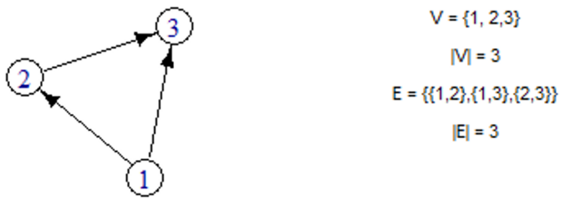

Figure 2: Directed simple graph Example

Neighbor

A neighbor of node i, is all nodes that node i are connected.

Adjacency Matrix

Apart from representing a graph diagrammatically, matrices are also one commonly used representation form of graphs. Let G represent a graph with n nodes ordered from v_1 to v_n. Then the adjacency matrix $A(G)$ is a $n \times n$ matrix in which the entry in row i and column j or a_{ij} indicates the situation that whether the node i is connected to node j:

$$a_{ij} = \begin{cases} 1, & i \text{ is connected with } j \\ 0, & \text{otherwise} \end{cases}$$

For instance, on the figure above (Fig 1.1), we have the graph with 8 nodes with its adjacency matrix, when two nodes are connected, the entry would be assigned to 1, otherwise it would be assigned to 0.

$$A_1 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Besides, if the node i is directly connected with j, the adjacency matrix of the graph would not be symmetrical about the main diagonal, like for the graph in Fig. 2, the adjacency matrix is:

$$A_2 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Degree Matrix

In addition to adjacency matrix, degree matrix is also used to present the characteristics of graphs. It is one a diagonal matrix which indicates the degree of each node. Degree of one node is the number of edges attached to this node. For directed graph, the nodes will have two degrees which are in-degree and out-degree. In-degree value is the number of incoming edges, respectively out-degree is the number of outgoing edges. The degree matrix D for G is a nxn diagonal matrix defined as:

$$D_{ij} = \begin{cases} deg(v_i), & \text{if} i = j \\ 0, & \text{otherwise} \end{cases}$$

For the graph of Fig. 1, the degree matrix is:

$$D_1 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

For the graph of Fig. 2, since it is one directed graph, it has two degree matrices including in-degree matrix and out-degree matrix,

$$D_{in} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

$$D_{out} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Laplacian Matrix

Laplacian matrix is another matrix used to represent the graph. It is defined by the difference between adjacency matrix and degree matrix. It is one very useful tool which can contribute to define many important properties of graph. Given a simple graph G with n nodes, the Laplacian matrix L_{nxn} is defined as: $L = D - A$ The elements of L are given by

$$L_{ij} = \begin{cases} deg(v_i), & \text{if} i = j \\ -1, & \text{if} i \neq j \text{and} v_i \text{is adjacent to} v_j 0, \\ \text{otherwise} \end{cases}$$

For the graph is Fig 1.1, the Laplacian matrix is: $L = D - A =$

$$L_1 = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

For the graph in Fig. 2, the Laplacian matrix is defined as:

$$L_{in} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & -1 & 1 \\ 0 & 0 & -2 \end{pmatrix}$$

$$L_{out} = \begin{pmatrix} -2 & 1 & 1 \\ 0 & -1 & 1 \\ 0 & 0 & -2 \end{pmatrix}$$

Degree Distribution

In the section of degree matrix, the degree of one node has been indicated as the number of connections the node owns. And the degree distribution is the probability distribution of degree for all the nodes in the network. Given a simple graph G with n nodes, if there exists n_k nodes with degree k, then the degree distribution $P(k)$ is presented as:

$$P(k) = \frac{n_k}{n} \tag{1}$$

For the above graph Fig. 1, all the non-zero elements are same which is 2, thus the degree distribution $P(2) = 1$. For the Fig. 1, the in-degree distribution and out-degree distribution are same, which contains: $P(1) = 0.5, P(2) = 0.5$.

2. Random Network Model

Random network is the graph with randomly generated nodes and edges, it is used in mathematical and computer science to extend the properties to general. There are many different types of random generation model, in this section the most famous and important random graph model refers to the Erdos-Renyi random model, scale free network and Watts-Strogatz Small-world will be described.

2.1 Erdos-Renyi random models

In mathematical context, the random graph is almost exclusively to the Erdos-Renyi random model, which is also the simplest random network model. The most important assumption of Erdos-Renyi is all the generations of random variables are independently. Following with this rule, ER random networks are the natural type of random networks. Due to this rule, it also brings the result that the degree distribution of the generated random network structure is very tight[2]. Erdos-Renyi brought up two graph models, today people usually use the G(n,p) model which is constructed through connecting nodes randomly. The probability for one edge to be involved in the edge is defined as p, and the probability for one graph to obtain n nodes and M edges is defined as[1]:

$$P^M(1-p)^{\binom{n}{2}-M} \tag{2}$$

Here is one simple example graph generated by Erdos-Renyi random model:

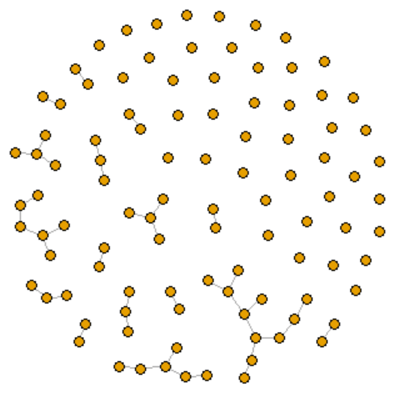

Figure 3: Erdos-Renyi Random Model Example

Scale Free Network

Scale-free model is designed to fit the property of practical world network which exist some nodes are highly connected to all the other nodes across the network, named hubs. The main characteristic of scale-free network is the power-law degree distribution, which means for all scales the power law can have the same functional form, when k is changed the degree distribution function $P(k)$ remains unchanged. For a scale-free undirected network, the degree distribution function P(k) follows that:

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

One example of scale-free network is as shown in the Fig. 4

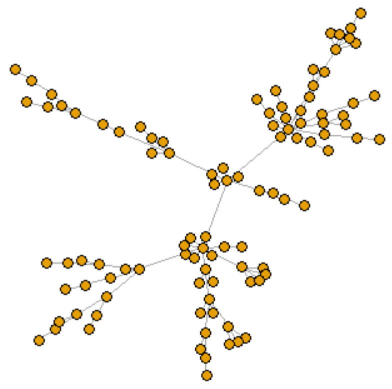

Figure 4: Scale Free Network Model Example

Watts-Strogatz Small-world

The small world property is defined as the path between two nodes involves only several edges, which is also presented in the simple Erdos-Renyi random model. However, in real life the network is much more complex, the degree of transitivity in network can be very large. Based on this, Watt and Strogatz constructed one model which combines the high transitivity with random network, named as small-world model. They developed the lattice model with high clustering and high transitivity through assigning one probability p that one edge is rewired to the model, which means the edge is disconnected from one of its nodes and then randomly connect to another node in the network. When the probability p becomes large, the small-world will become similar to the ER random network model[4] . One example of Watts-Strogatz Small-world network model is shown in Fig. 5.

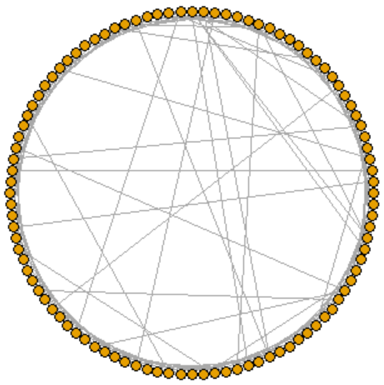

Figure 5: Watts-Strogatz Small-world Network Model Example

Chapter 2: Distributed Average Algorithm

After the basic introduction about graph, one important item of graph which is consensus will be the focus of this section. Achieving average consensus is very important in network model construction, the faster the consensus is, the better the performance of the network. [5] In this report, the distributed average consensus algorithm for graph will be considered. The basic idea of average consensus will be illustrated in this section, applied with the realization of different graph models to observe the algorithm.

1. Distributed Average Consensus Model

Given the graph $G = (V, E)$ as one undirected graph, where V is the nodes set and E is the edge set. Assume $x_i(0)$ is one quantities assigned to the node at time t=0, the distributed average consensus problem for the graph is calculating the value of $\frac{1}{n} sum_{i=1}^{n} x_i 0$ for all the nodes in the graph[6]. At each step, each node carries out its update based on its local state and communication with its direct neighbors. More precisely, the difference equations for the network at time k can be expressed as[7].:

$$X_i(k+1) = \frac{1}{1+d_i}(x_i(k) + \sum_{j_i} x_j(k)) \tag{3}$$

Where N_i is the neighbor of node i, and d_i denotes the degree of node i.

2. Consensus Results Visualization

To better understand the idea of distributed average consensus algorithm, such different types of graphs will be applied to observe the whether the consensus equilibrium state can be achieved at the average consensus.

2.1 Simple graph

We start from simple graph examples at Fig 1 and Fig 2. The results are shown in Fig 6 and 7.

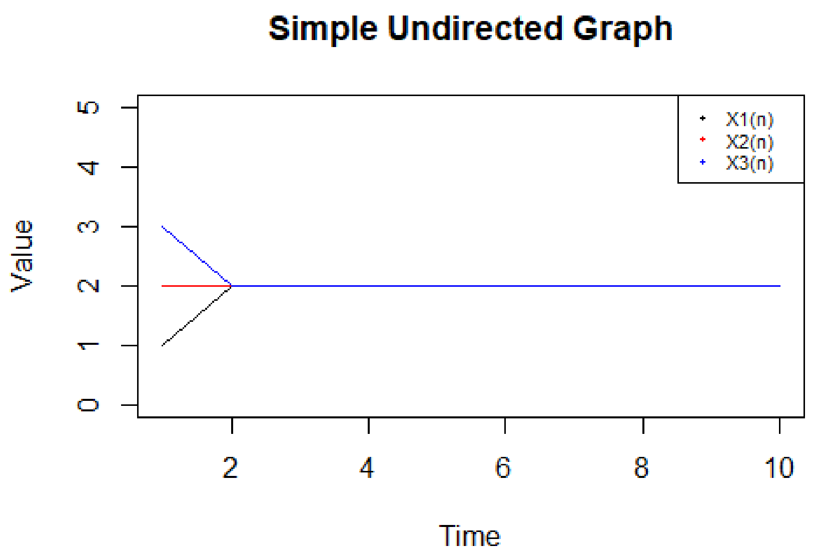

Figure 6: Consensus for Simple Undirected Graph

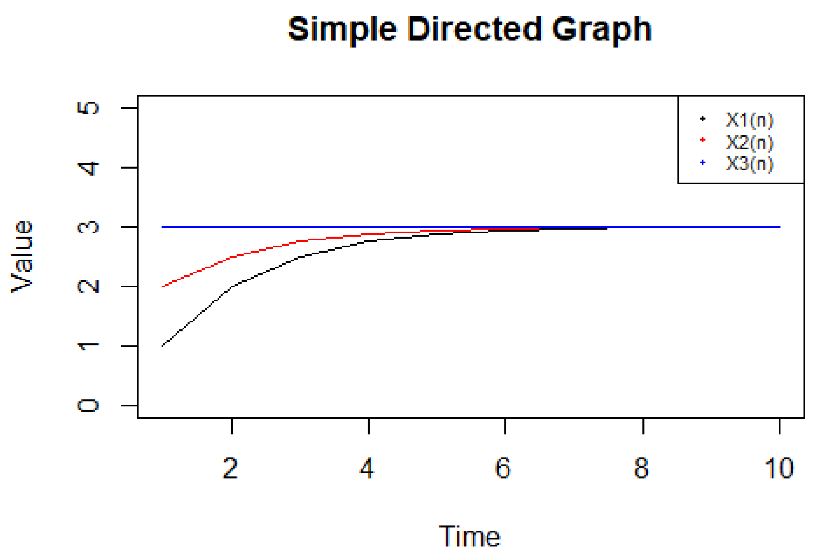

Figure 7: Consensus for Simple Directed Graph

The visualization results indicates that both these two graph have reached equilibrium state at the average consensus.

2.2 Complex graph

In this section the more complex graphs than the above networks will be applied. The undirected and directed graph which will be applied here are shown in Fig 8 and 9. Same as the simple networks in Chapter 1, the mathematical expression of the graphs will be posted on the right part of figure.

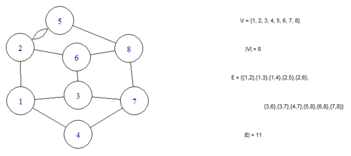
5
2
6
8
1
3
7
4
V = {1, 2, 3, 4, 5, 6, 7, 8}
|V| = 8
E = {{1,2},{1,3},{1,4},{2,5},{2,6},
{3,6},{3,7},{4,7},{5,8},{6,8},{7,8}}
|E| = 11

Figure 8: Undirected complex graph Example

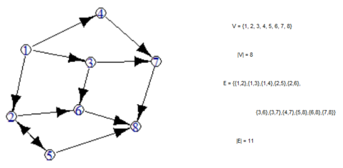
4
1
3
7
2
6
8
5
V = {1, 2, 3, 4, 5, 6, 7, 8}
|V| = 8
E = {{1,2},{1,3},{1,4},{2,5},{2,6},
{3,6},{3,7},{4,7},{5,8},{6,8},{7,8}}
|E| = 11

Figure 9: Directed complex graph Example

The graphs used here contains 8 nodes, and they are connected graph. The visualization results of average consensus are shown in the following graphs.

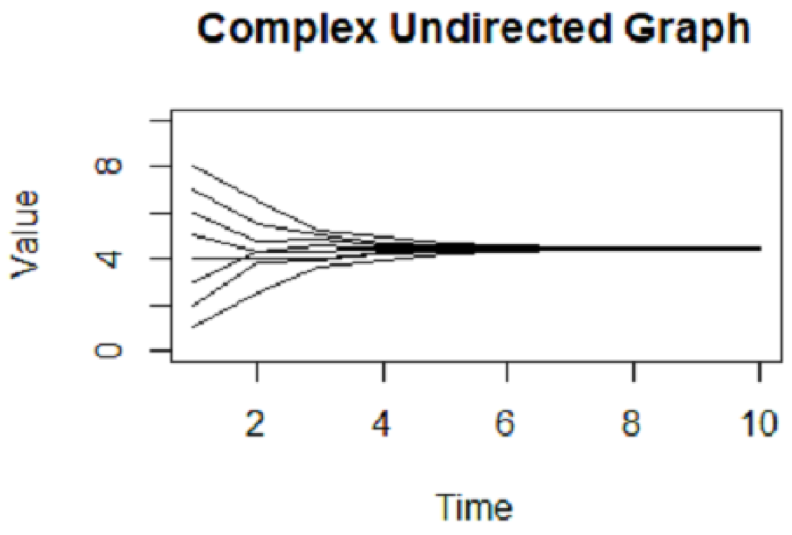

Figure 10: Consensus for Complex Undirected Graph

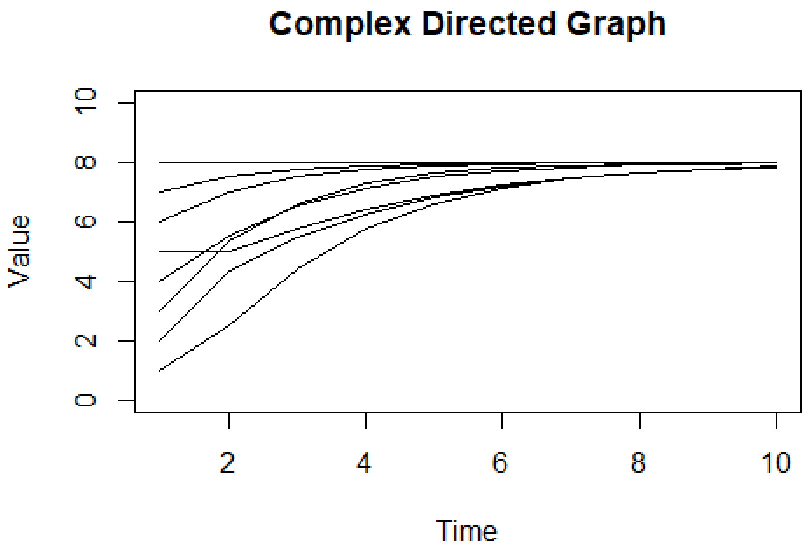

Figure 11: Consensus for Complex Directed Graph

The visualization results indicates that both these two graph have reached equilibrium state at the average consensus.

2.3 Random Network

In this section, the three random network model mentioned before will be realized and applied with the distributed average consensus to observe whether they could reach the equilibrium state at average consensus. Due to the large scale of random network data, only the first ten columns have been presented on the plots.

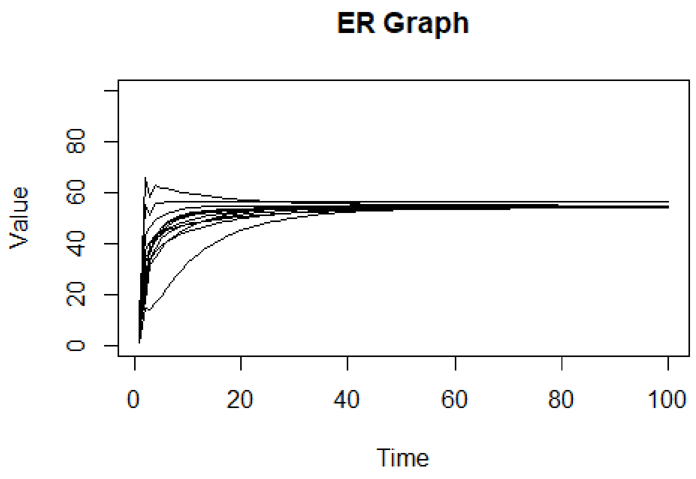

Figure 12: Consensus for Erdos-Renyi random model

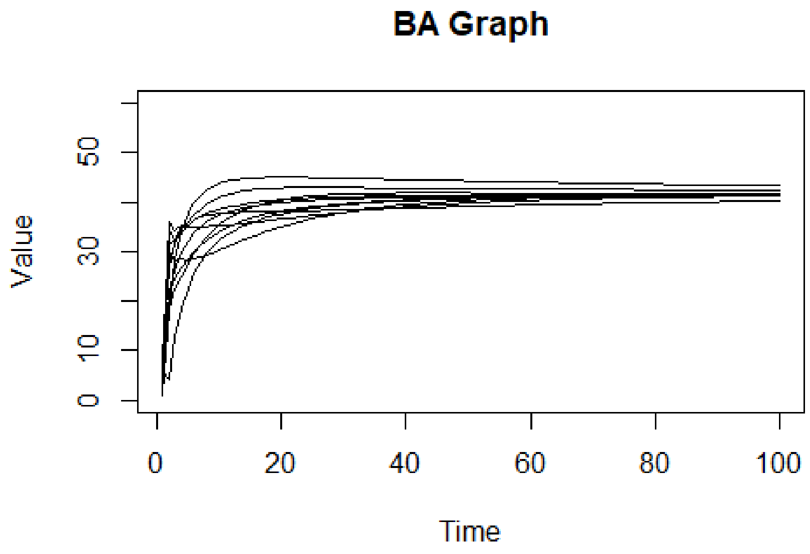

Figure 13: Consensus for Scale-free Networks

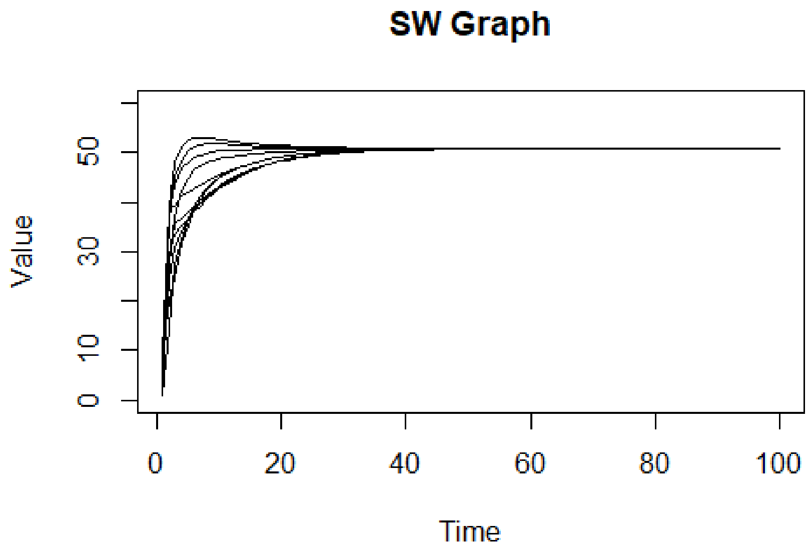

Figure 14: Consensus for Watts-Strogatz Small-world Networks

The visualization results indicates that all these three random network model have reached equilibrium state at the average consensus.

Chapter 3: Conclusion

In this report we have obtained several different types of graphs to test the distributed consensus algorithm, and obtained the result that all the models can reach equilibrium state at the average consensus, Average consensus algorithms consist of computing the average of the initial state values in a distributed fashion, proved that all the network models guaranteed convergence with the algorithm. Applications of average consensus algorithms can be found in many areas, it is very important to keep the research of consensus algorithms about networks.

REFERENCES

References

[1] Erd P and R A. On Random Graphs. I. Publicationes Mathematicae, 6:290-297, 1959.

[2] Nykamp DQ, "One of the simplest types of networks." From Math Insight. http://mathinsight.org/simplest type of network

[3] Barab AL, Albert R. Statistical mechanics of complex networks. Rev. Modern Physics, (7) (2002) 47-97

[4] Watts DJ and Strogatz SH. Collective dynamics of 'small-world' networks. Nature,(1998) 393:440-442

[5] Deming Yuan, Shengyuan Xu, Huanyu Zhao, Yuming Chu, Distributed average consensus via gossip algorithm with real-valued and quantized data for 0<q<1, Systems Control Letters, Volume 59 (9) (2010) 536-542

[6] Lin Xiao, Stephen Boyd, Seung-Jean Kim, Distributed average consensus with least-mean-square deviation, Journal of Parallel and Distributed Computing, Volume 67 (1) (2007) 33-46

[7] JALILI, M. A simple consensus algorithm for distributed averaging in random geographical networks. Pramana - J Phys 79, 493–499 (2012).

Appenix

R Environment

```
library(igraph)
```

Algorithms

```
#distributed average consensus algorithm
#n: number of nodes in the graph
d=matrix(1:n,ncol=1)
#calculate the adjacency matrix of the graph
getM <- function(x,n){
  A <-x[1]
  for (i in 2:n) {
    A <- rbind(A,x[i])
  }
  return(A)
}
A <- getM(n,100)

for (i in 1:ncol(A)) {
  A[i,i] <- A[i,i]+1
}

sumA=rowSums(A)

f <- function(k){
  d <- cbind(d,c(rep(0,n)))
  m <- matrix(d[,k-1], ncol = 1)
  x <- (A%*%m)/sumA
  d[,k] <- x
  return(d)
}

for (i in 2:n) {
  d <- f(i)
  d <- d
}

plot(1:n,d[1,], type = "l", ylim = c(0,60), xlab = "Time", ylab = "Value",main="Graph")
lines(1:100,d[2,])
lines(1:100,d[3,])
legend("topright",pch=20,legend = c("X1(n)","X2(n)","X3(n)")
, col = c("black","red","blue"),cex=0.7)
```

Figure

- Fig 1

  ```
  #simple undirected graph
  layout(rbind(c(1,2)), respect = T)
  ug = graph.ring(3)
  as.undirected(ug)
  plot(ug,vertex.color="white",vertex.size = 50,edge.color="black",margin = rep(0, 4))
  plot(0:5, 0:5, type = "n", xlab = "", ylab = "", main = "", axes = F)
  text(1, 5, "V␣=␣{1,␣2,3}", cex = 0.7)
  text(1, 4, "|V|␣=␣3", cex = 0.7)
  text(1, 3, "E␣=␣{{1,2},{1,3},{2,3}}", cex = 0.7)
  text(1, 2, "|E|␣=␣3", cex = 0.7)
  ```

- Fig 2

  ```
  #simple directed graph
  dg = make_graph(~1:2--+2:3)
  plot(dg,vertex.color="white",vertex.size = 50,edge.color="black",margin = rep(0, 4))
  ```

```
plot(0:5, 0:5, type = "n", xlab = "", ylab = "", main = "", axes = F)
text(1, 5, "V␣=␣{1,␣2,3}", cex = 0.7)
text(1, 4, "|V|␣=␣3", cex = 0.7)
text(1, 3, "E␣=␣{{1,2},{1,3},{2,3}}", cex = 0.7)
text(1, 2, "|E|␣=␣3", cex = 0.7)
```

- Fig 3

```
#Erdos-Renyi random model
er <- sample_gnm(n=100, m=40)
plot(er, vertex.size=6, vertex.label=NA)
```

- Fig 4

```
#Scale-free Networks
ba <-  sample_pa(n=100, power=1, m=1,  directed=F)
plot(ba, vertex.size=6, vertex.label=NA)
```

- Fig 5

```
#Watts-Strogatz Small-world
sw <- sample_smallworld(dim=2, size=10, nei=1, p=0.1)
plot(sw, vertex.size=6, vertex.label=NA, layout=layout_in_circle)
```

- Fig 8

```
#complex undirected graph
m = matrix( c(0,1,5, 0,2,4, 0,3,3, 1,5,3, 1,4,5, 2,5,3, 2,6,2,
3,6,2, 4,1,5, 4,7,4, 5,7,3, 6,7,5) + 1,ncol = 3,byrow = T)
ug = make_graph(t(m[,1:2]),directed = FALSE)
plot(ug,vertex.color="white",vertex.size = 50,edge.color="black",margin = rep(0, 4))
plot(0:10, 0:10, type = "n", xlab = "", ylab = "", main = "", axes = F)
text(2, 9, "V␣=␣{1,␣2,␣3,␣4,␣5,␣6,␣7,␣8}", cex = 0.7)
text(1, 7, "|V|␣=␣8", cex = 0.7)
text(2, 5, "E␣=␣{{1,2},{1,3},{1,4},{2,5},{2,6},", cex = 0.7)
text(4, 3, "{3,6},{3,7},{4,7},{5,8},{6,8},{7,8}}",cex = 0.7)
text(1, 1, "|E|␣=␣11", cex = 0.7)
```

- Fig 9

```
#complex directed graph
dg = make_graph(t(m[,1:2]),directed = TRUE)
plot(dg, vertex.color="white",edge.color = "black")
plot(0:10, 0:10, type = "n", xlab = "", ylab = "", main = "", axes = F)
text(2, 9, "V␣=␣{1,␣2,␣3,␣4,␣5,␣6,␣7,␣8}", cex = 0.7)
text(1, 7, "|V|␣=␣8", cex = 0.7)
text(2, 5, "E␣=␣{{1,2},{1,3},{1,4},{2,5},{2,6},", cex = 0.7)
text(4, 3, "{3,6},{3,7},{4,7},{5,8},{6,8},{7,8}}",cex = 0.7)
text(1, 1, "|E|␣=␣11", cex = 0.7)
```