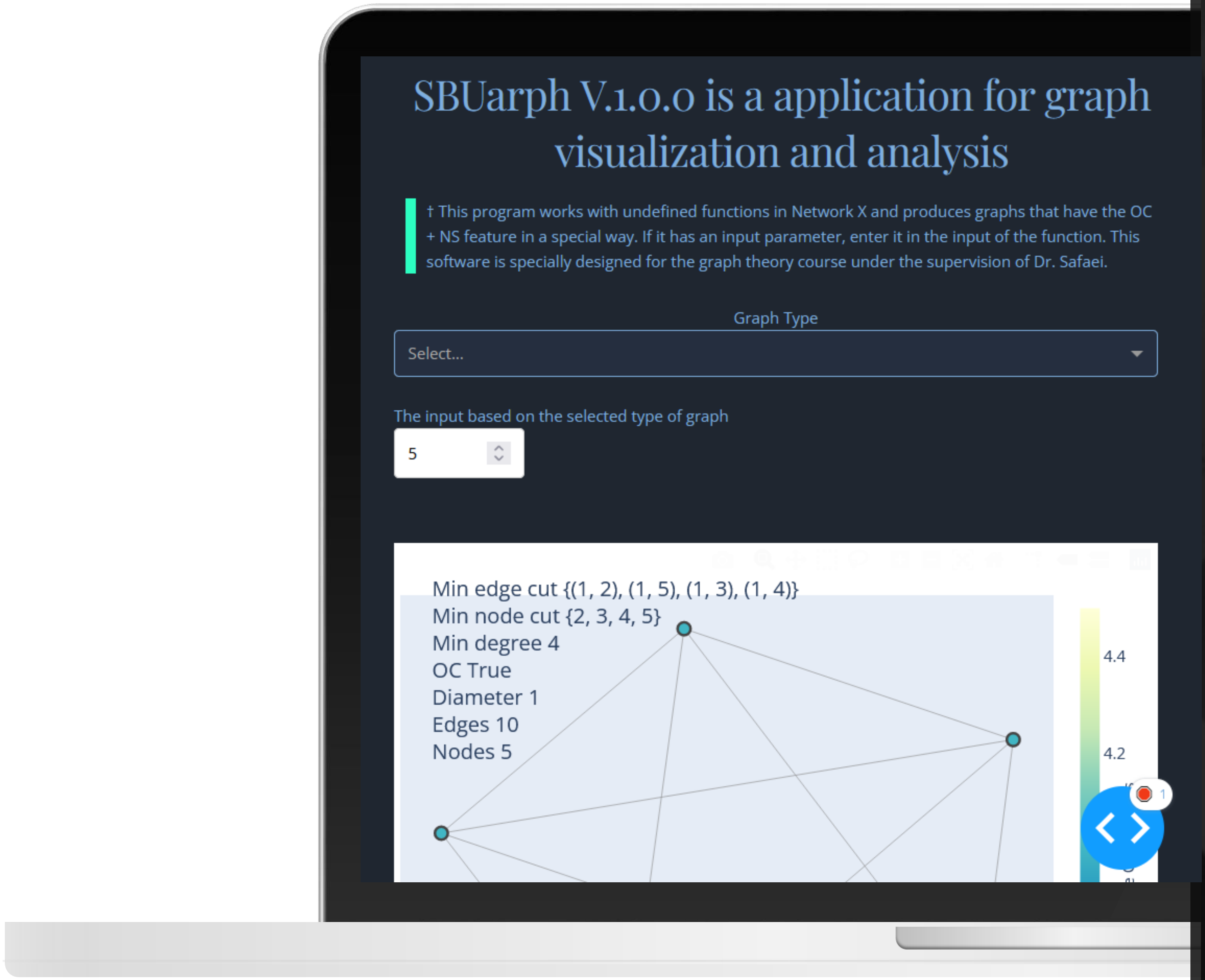


SBUarph V.1.0.0 is a application for graph visualization and analysis

† This program works with undefined functions in Network X and produces graphs that have the OC + NS feature in a special way. If it has an input parameter, enter it in the input of the function. This software is specially designed for the graph theory course under the supervision of Dr. Safaei.

Graph Type: Select...

The input based on the selected type of graph: 5

```
Min edge cut {(1, 2), (1, 5), (1, 3), (1, 4)}
Min node cut {2, 3, 4, 5}
Min degree 4
OC True
Diameter 1
Edges 10
Nodes 5
```

Rashin Rahnamoun
Kiarash Kowsari

Final Graph Theory Course Project Report

Dr.Safaei

REPORT MANUAL

1

Basic information

All the questions are written with the number and the final result, and all the results are obtained non-manually and completely by software in order to avoid errors as much as possible.

2

SBUarph application

It is hoped that the SBUarph software will solve the shortcomings of Network X and after analyzing and checking the results of all the tools and simulations, if there are no problems, it will be converted into a web-based software so that everyone can check the graphs without having any programming knowledge. be simple

3

Future plans

After checking the results and making sure of their output and correction, we hope that a complete version of this software will be available to everyone. With an ambitious opinion, maybe it can be a competitor to Network X and compensate for its shortcomings, because Python programming is allowed in it and it is not separated from the graphical interface, and it can be available to everyone in the form of open source.

Report

1

Minimum Spanning Tree: A minimum spanning tree (MST) is a subgraph of a connected, undirected graph that includes all the vertices of the original graph while minimizing the total weight of the edges. An MST ensures optimal connectivity by providing a tree-like structure that spans all vertices with the minimum possible edge weight.

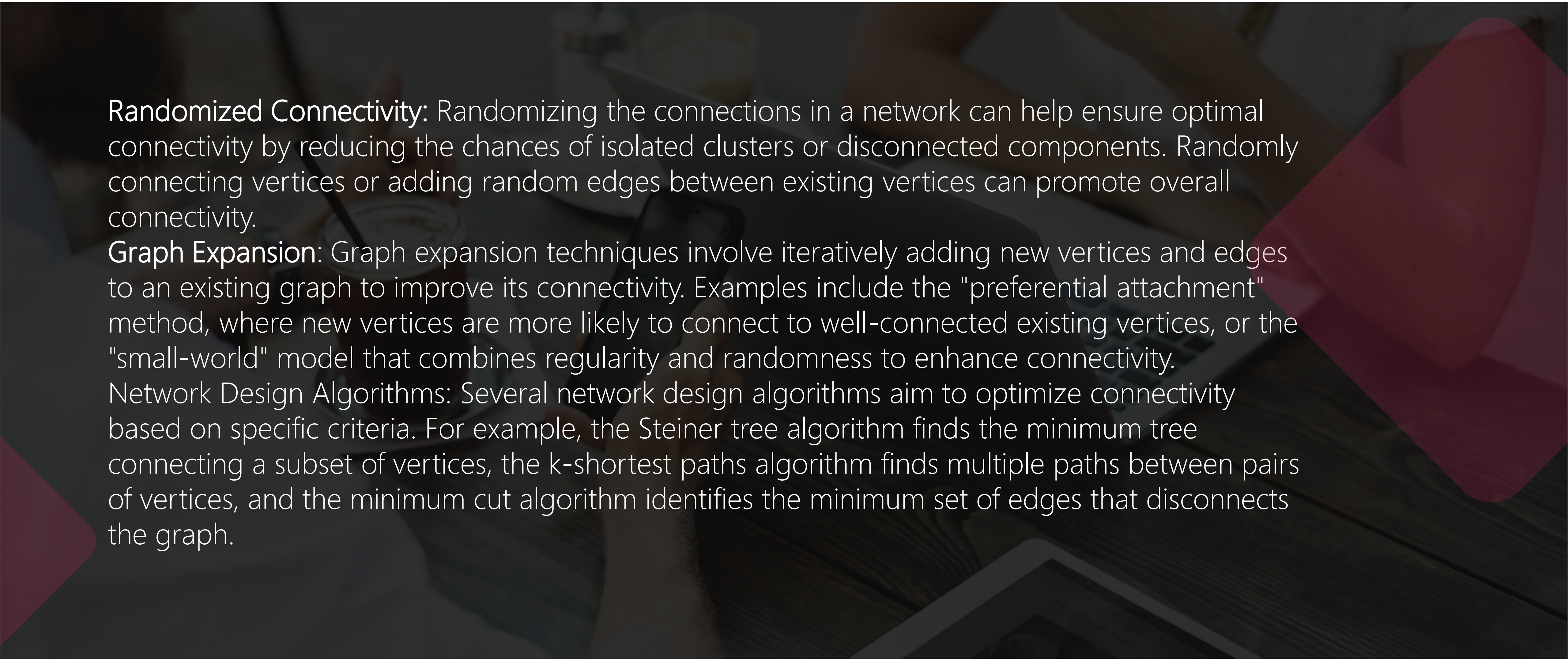
Maximal Connected Subgraph: Finding a maximal connected subgraph involves identifying the largest possible connected subgraph within a given graph. It aims to maximize the number of vertices connected to each other, promoting optimal connectivity.

Connectivity Algorithms: Various algorithms exist to optimize connectivity in graphs, such as the breadth-first search (BFS) or depth-first search (DFS) algorithms. These algorithms help identify the connected components within a graph, determine if a graph is connected, or find the shortest path between vertices, enhancing the overall connectivity.

And in Networks:

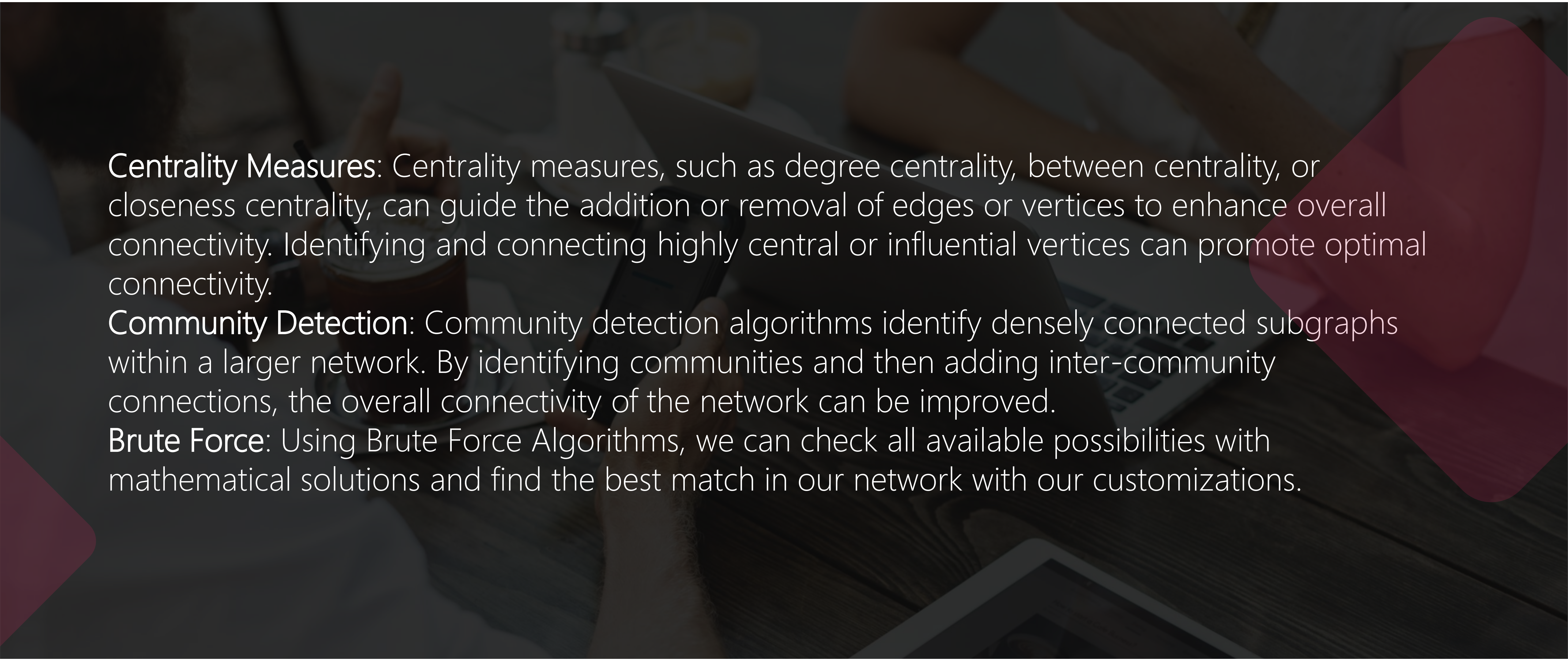
Complete Graph: A complete graph is a graph in which every pair of vertices is connected by an edge. Creating a complete graph ensures that there is a direct connection between every pair of vertices, maximizing the connectivity.

Report 1



Randomized Connectivity: Randomizing the connections in a network can help ensure optimal connectivity by reducing the chances of isolated clusters or disconnected components. Randomly connecting vertices or adding random edges between existing vertices can promote overall connectivity.
Graph Expansion: Graph expansion techniques involve iteratively adding new vertices and edges to an existing graph to improve its connectivity. Examples include the "preferential attachment" method, where new vertices are more likely to connect to well-connected existing vertices, or the "small-world" model that combines regularity and randomness to enhance connectivity.
Network Design Algorithms: Several network design algorithms aim to optimize connectivity based on specific criteria. For example, the Steiner tree algorithm finds the minimum tree connecting a subset of vertices, the k-shortest paths algorithm finds multiple paths between pairs of vertices, and the minimum cut algorithm identifies the minimum set of edges that disconnects the graph.

Report 1

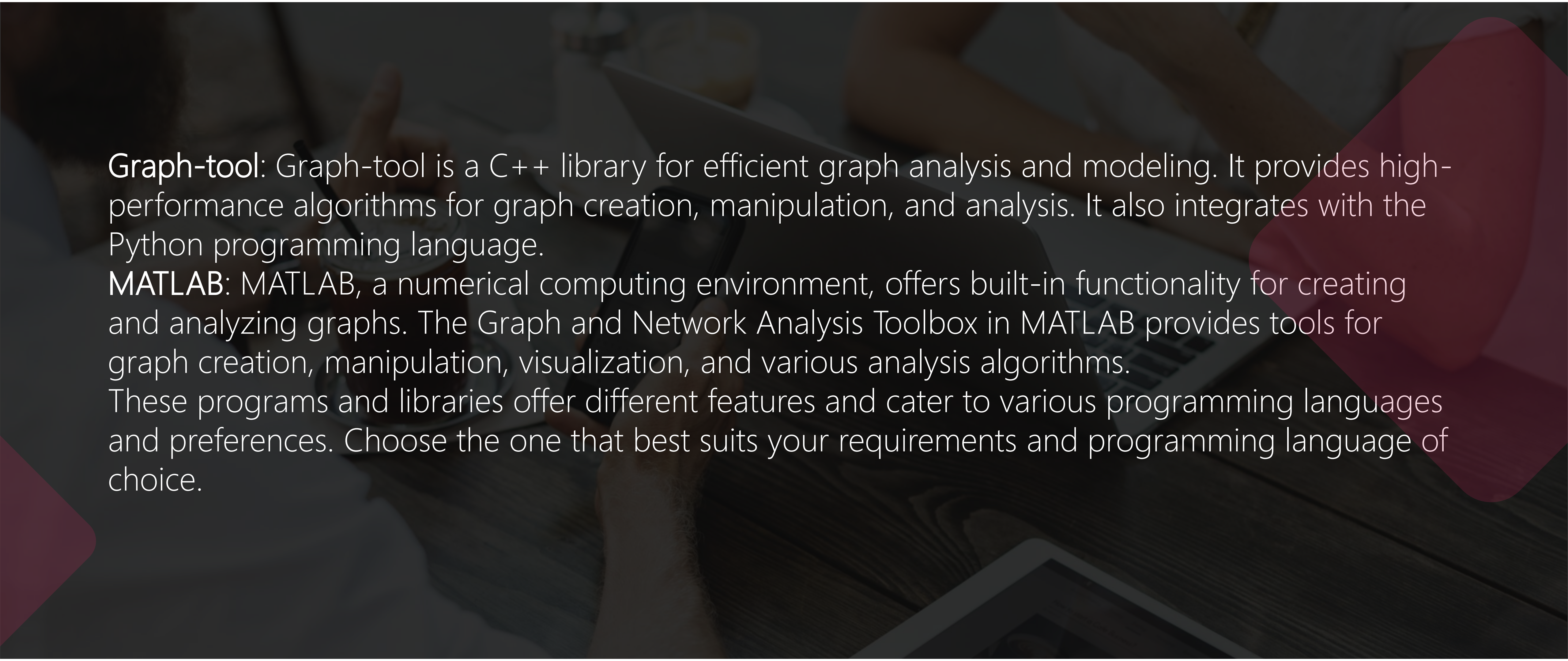


Centrality Measures: Centrality measures, such as degree centrality, between centrality, or closeness centrality, can guide the addition or removal of edges or vertices to enhance overall connectivity. Identifying and connecting highly central or influential vertices can promote optimal connectivity.
Community Detection: Community detection algorithms identify densely connected subgraphs within a larger network. By identifying communities and then adding inter-community connections, the overall connectivity of the network can be improved.
Brute Force: Using Brute Force Algorithms, we can check all available possibilities with mathematical solutions and find the best match in our network with our customizations.

Report 2

NetworkX: NetworkX is a Python library for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. It provides a wide range of graph creation functions, algorithms, and analysis tools.
igraph: igraph is a collection of network analysis tools available for several programming languages, including Python, R, and C/C++. It offers a comprehensive set of features for creating, manipulating, and analyzing graphs.
Gephi: Gephi is an open-source network visualization and exploration software. It provides an interactive interface for creating, exploring, and analyzing graphs. Gephi supports a variety of graph file formats and offers numerous layout algorithms and analysis plugins.
Cytoscape: Cytoscape is another popular open-source software platform for visualizing, analyzing, and modeling complex networks. It offers an extensive set of features for creating and analyzing graphs, including various layout algorithms and plugins.

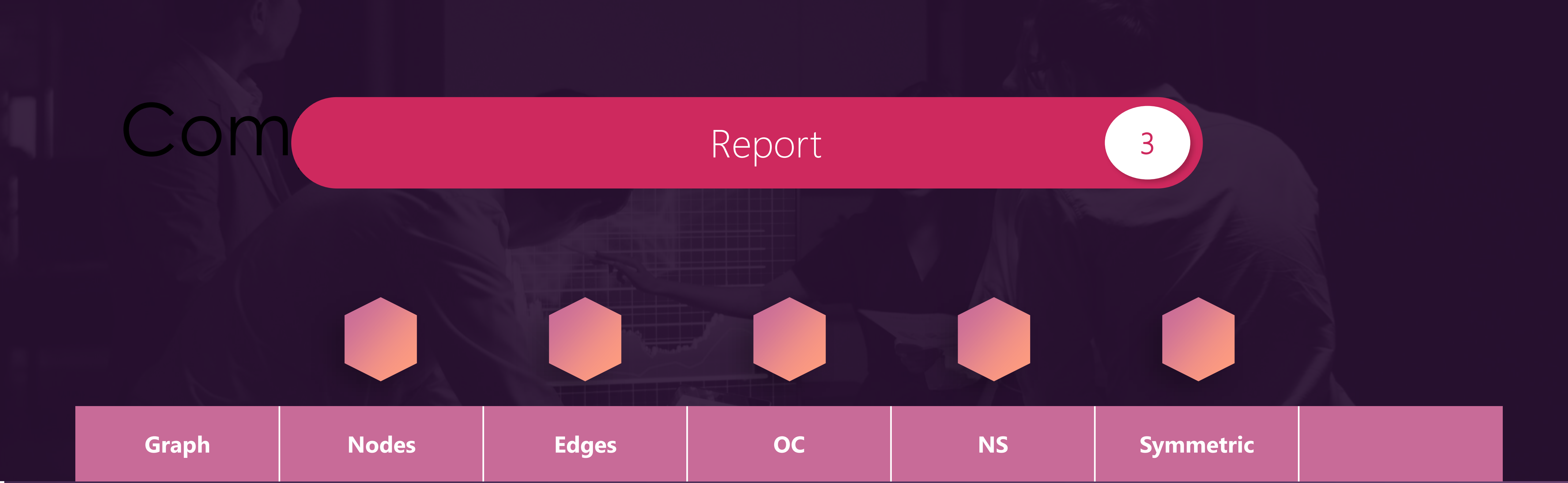
Report 2



Graph-tool: Graph-tool is a C++ library for efficient graph analysis and modeling. It provides high-performance algorithms for graph creation, manipulation, and analysis. It also integrates with the Python programming language.

MATLAB: MATLAB, a numerical computing environment, offers built-in functionality for creating and analyzing graphs. The Graph and Network Analysis Toolbox in MATLAB provides tools for graph creation, manipulation, visualization, and various analysis algorithms.

These programs and libraries offer different features and cater to various programming languages and preferences. Choose the one that best suits your requirements and programming language of choice.



Com

Report

3

Graph	Nodes	Edges	OC	NS	Symmetric	
tetrahedron	4	6	✓	✓	✓	
octahedron	6	12	✓	✓	✓	
cube	8	12	✓	✓	✓	
Icosahedron*	12	30	✓	✓	✓	

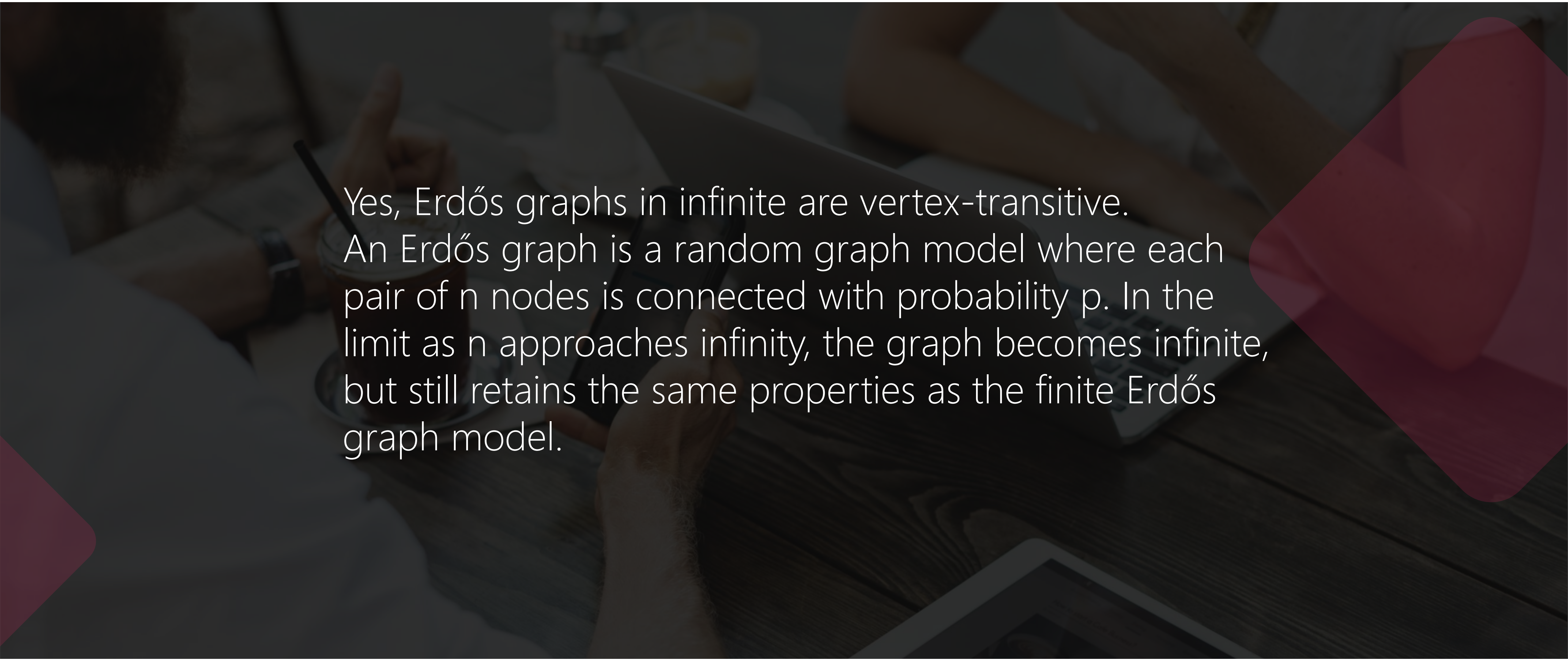
Com

Report

3

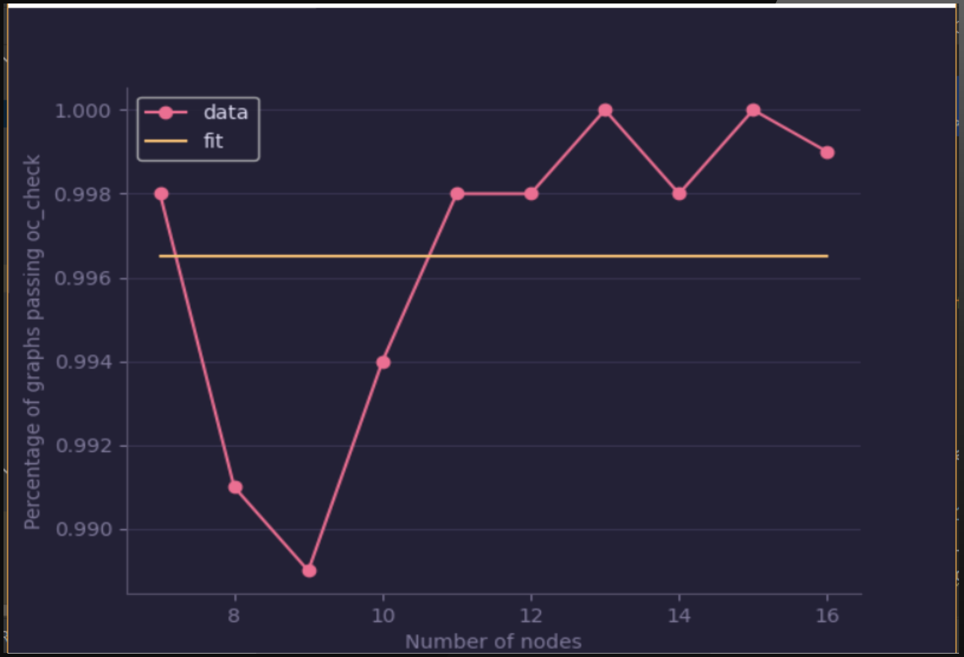
Graph	Nodes	Edges	OC	NS	Symmetric	
Soccer ball*	32	90	✓	✓	✓	
Truncated tetrahedron	12	18	✓	✓	✓	
Hypercube	4	4	✓	✓	✓	
Torus(grid)	4	4	✓	✓	✓	

Report 4



Yes, Erdős graphs in infinite are vertex-transitive.
An Erdős graph is a random graph model where each pair of n nodes is connected with probability p. In the limit as n approaches infinity, the graph becomes infinite, but still retains the same properties as the finite Erdős graph model.

For an infinite Erdős graph, the minimum node cut is equal to 1 almost surely. This means that for any two vertices in the graph, there is almost always a set of vertices, of size 1, that, if removed, disconnects the two vertices.
On the other hand, the minimum edge cut of an infinite Erdős graph is equal to 0 almost surely. This means that there is almost always a set of edges that, if removed, does not disconnect any two vertices. The simulation test graph are too small to show that after this chart in big numbers the percentage of OC connection will be zero. The fit needs more data two have good representation ☺



Com

Report 5

Graph	Nodes	Edges	OC	NS	Symmetric	diameter
Hypercube	4	4	✓	✓	✓	2
Torus(grid)*	4	4	✓	✓	✓	2
Ring	3	3	✓	✓	✓	1
Prism(note)	6	9	✓	✓	X	2

Com

Report 5

Graph	Nodes	Edges	OC	NS	Symmetric	diameter
Antiprism(note)	6	12	✓	✓	✓	2
Twisted prism	8	12	✓	✓	✓	2

CHART RESULTS

1

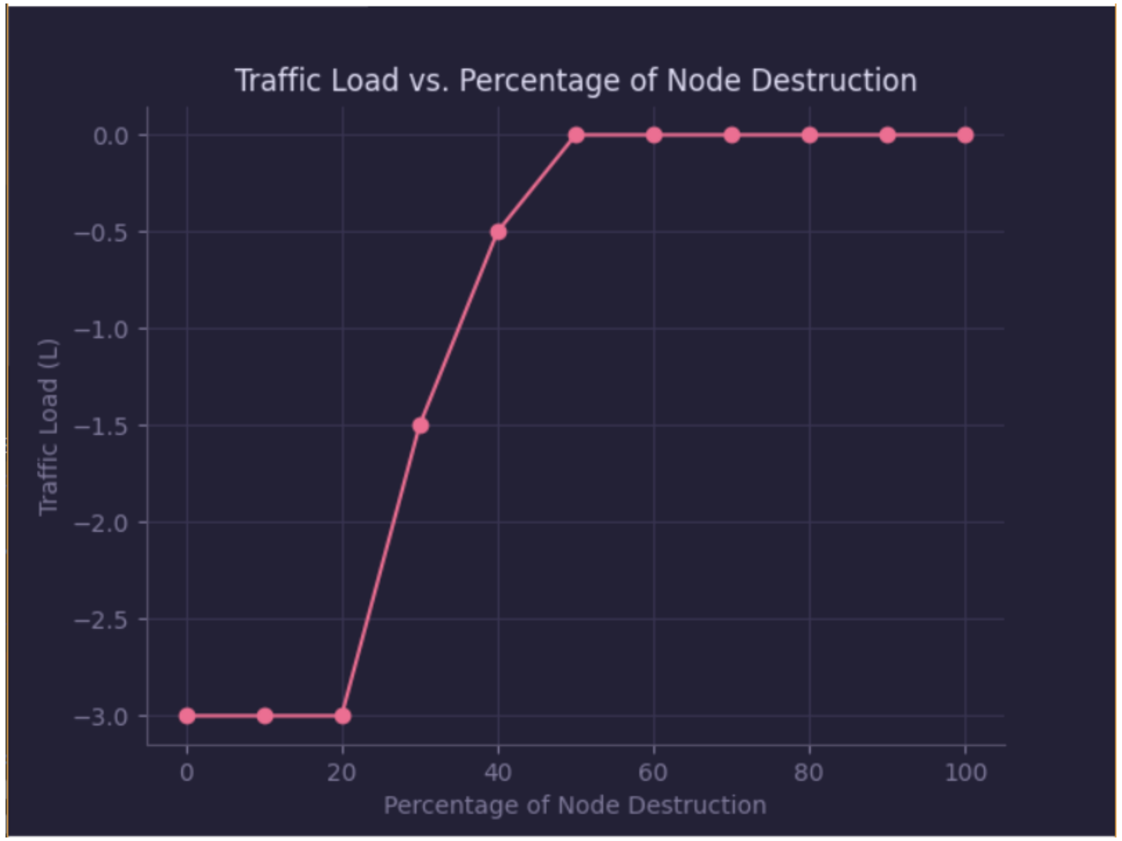


CHART RESULTS

2

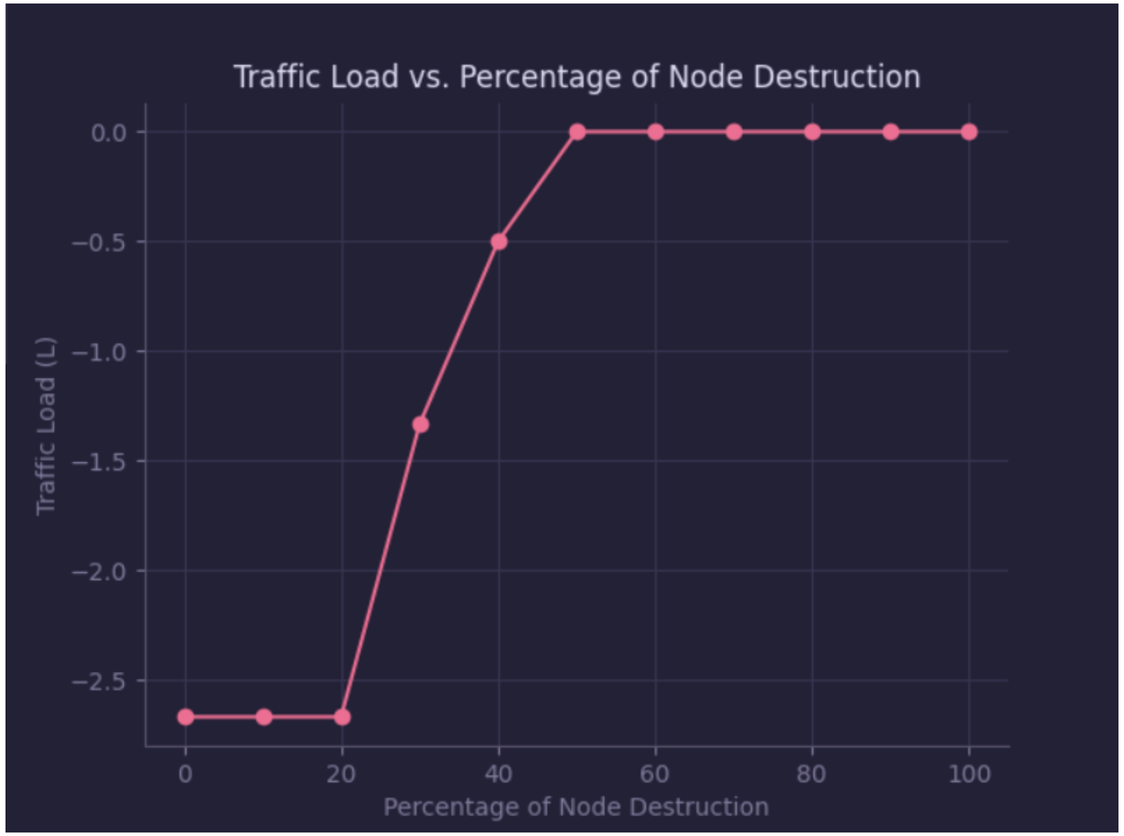


CHART RESULTS

3

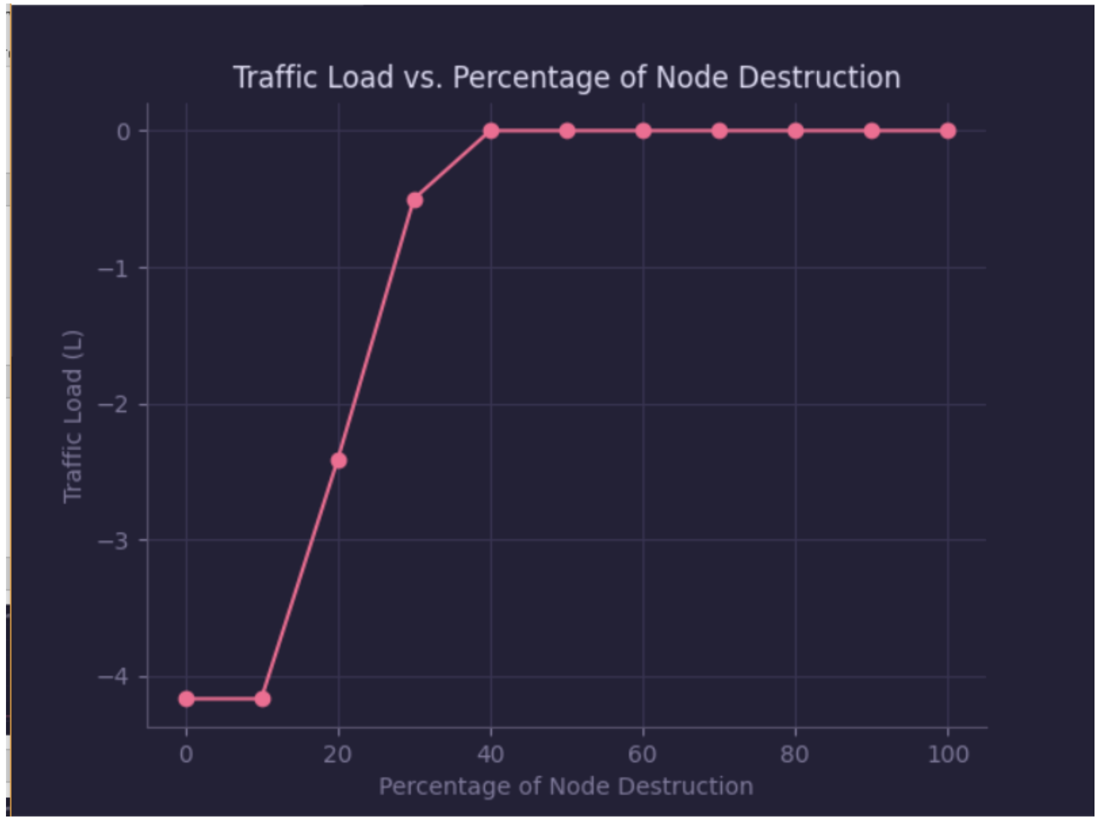


CHART RESULTS

4

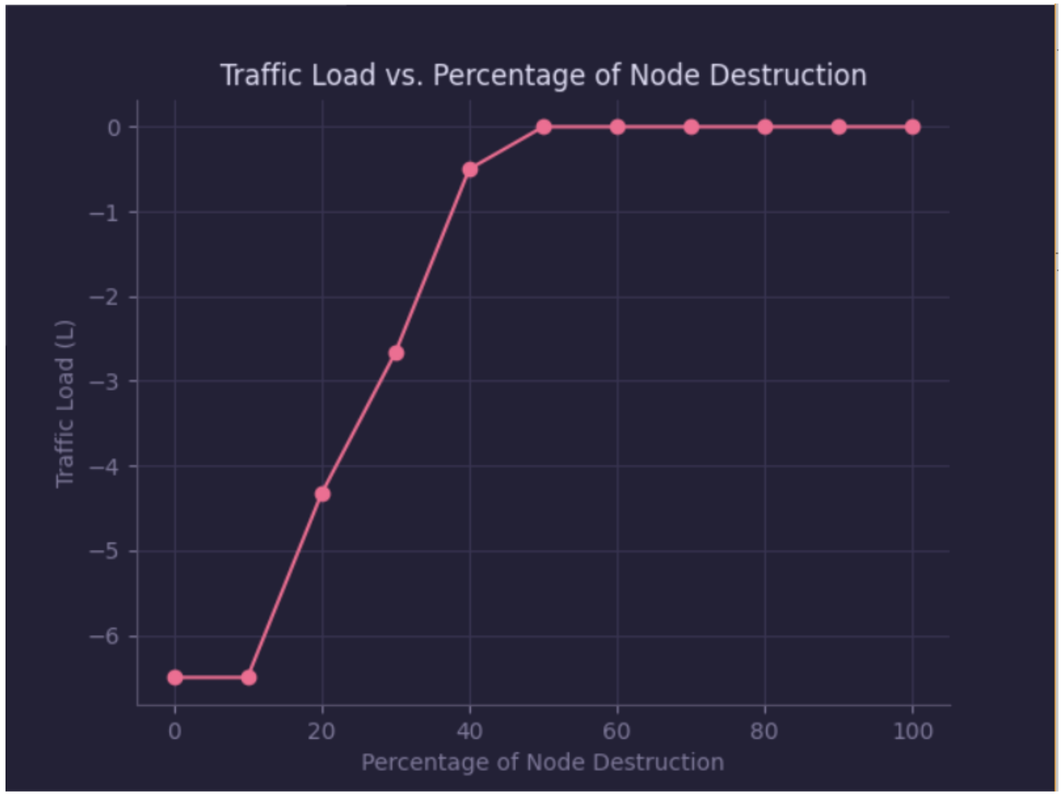


CHART RESULTS

5

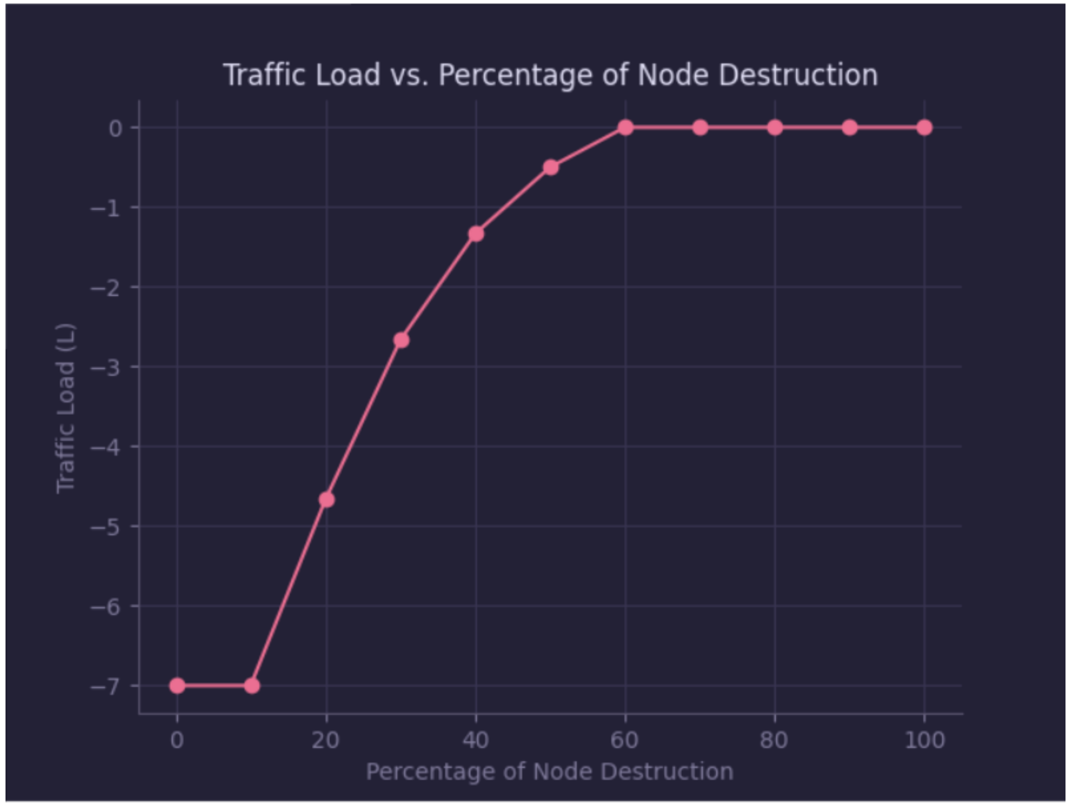
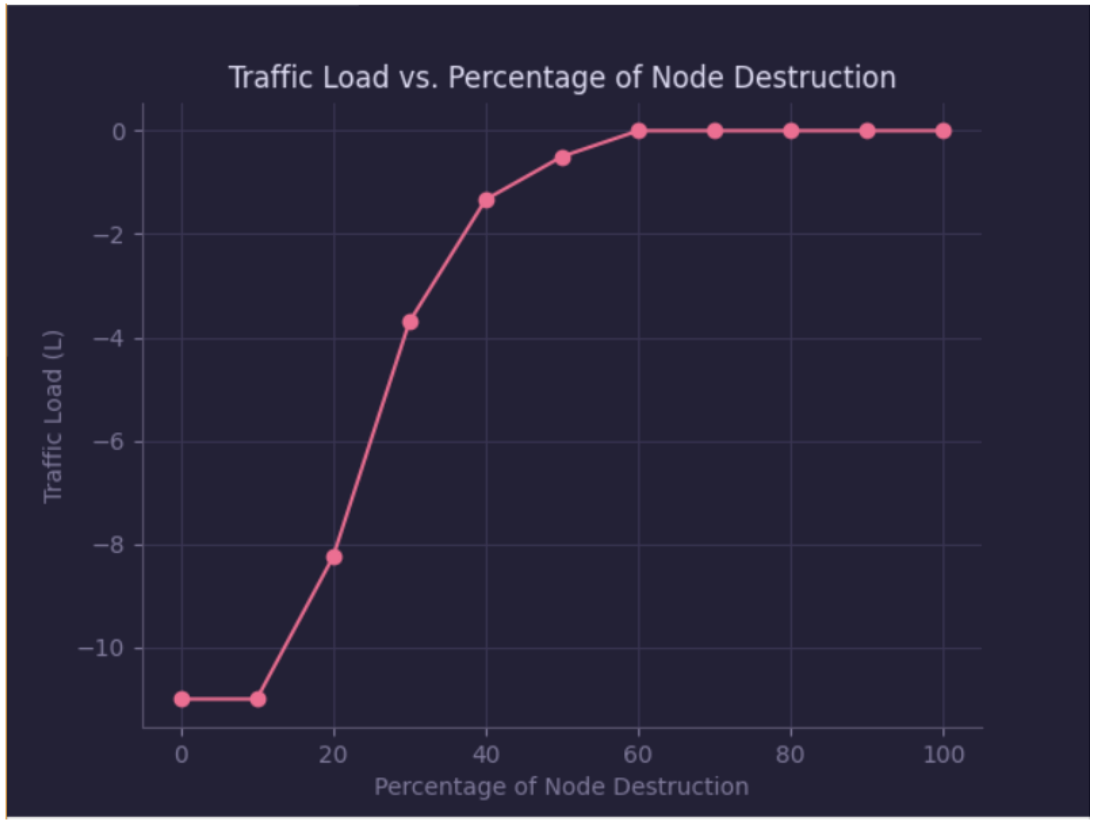
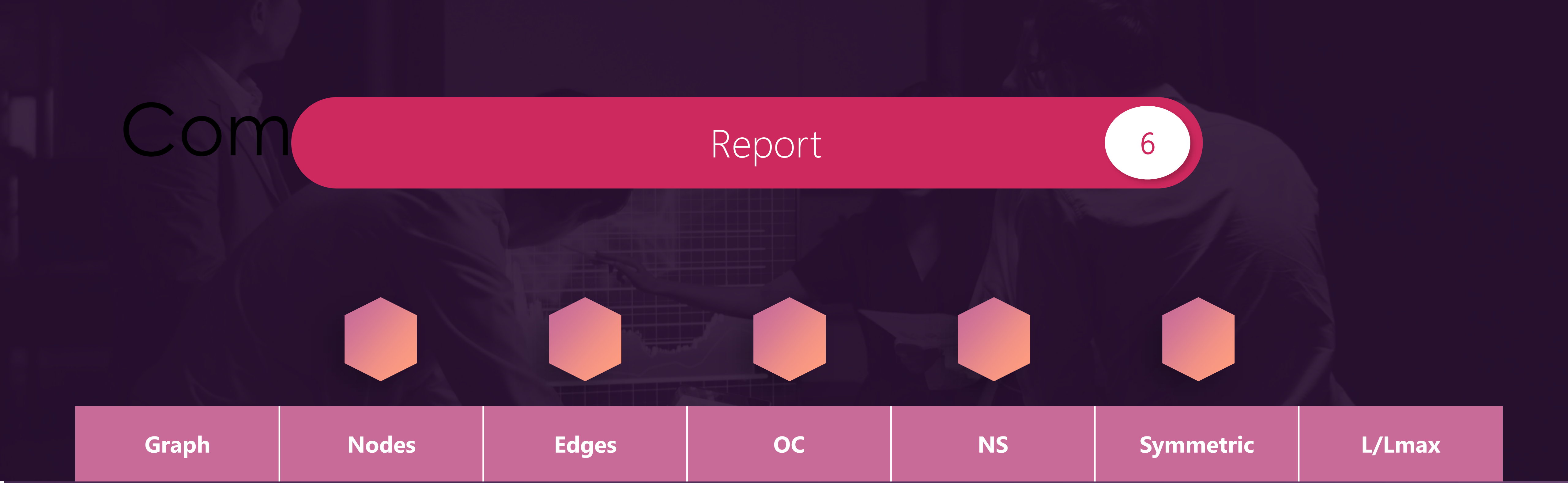


CHART RESULTS

6





Com

Report 6

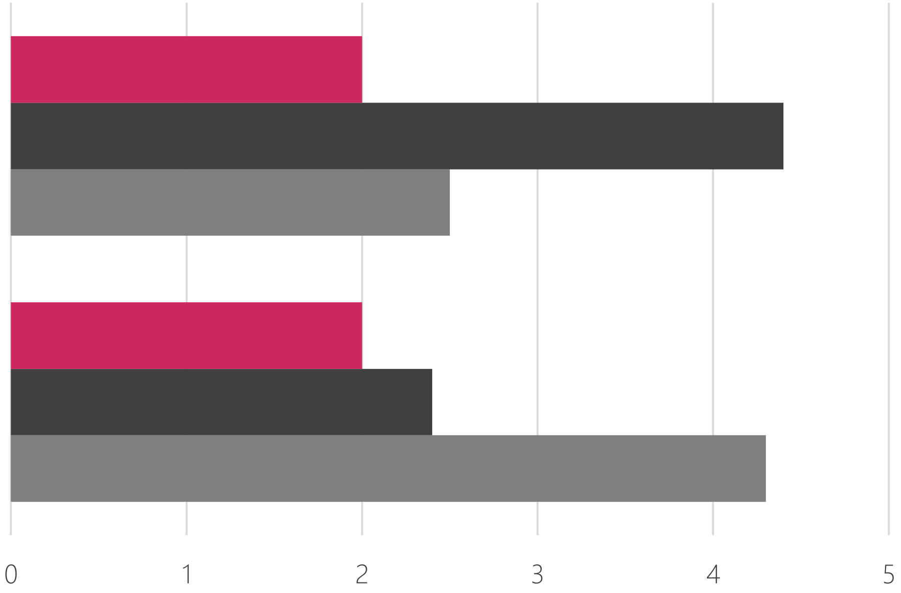
Graph	Nodes	Edges	OC	NS	Symmetric	L/Lmax
Kn	4	6	✓	✓	✓	3
Torus(grid)*	4	4	✓	✓	✓	2.6
Ring	5	5	✓	✓	✓	4.16
Prism(note)	6	9	✓	✓	X	6.5

Com

Report

6

Graph	Nodes	Edges	OC	NS	Symmetric	L/Lmax
Antiprism(note)	6	12	✓	✓	✓	7
Hamming	8	12	✓	✓	✓	11

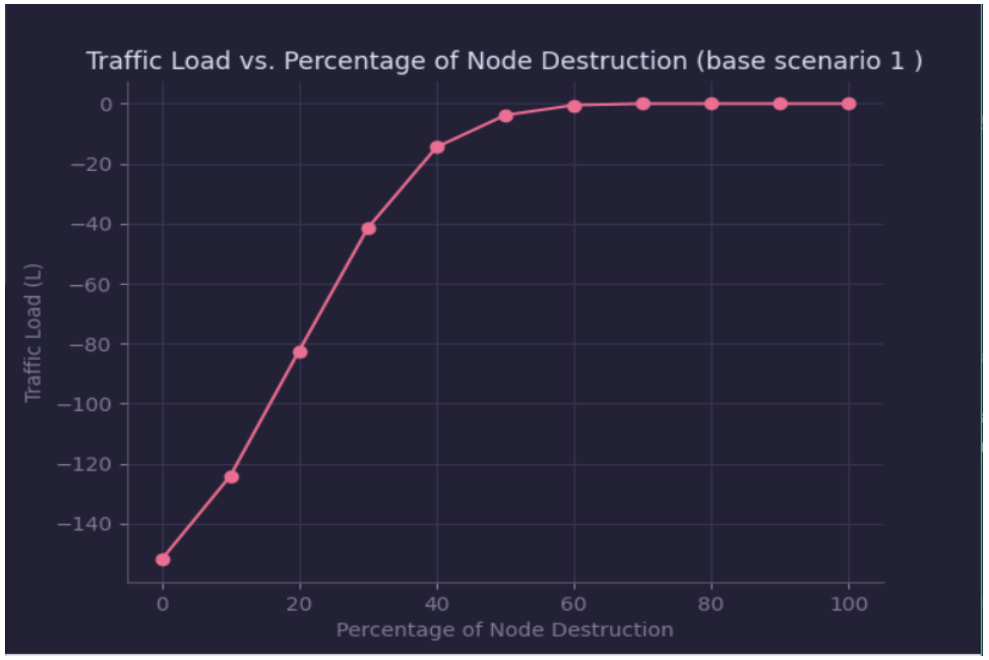


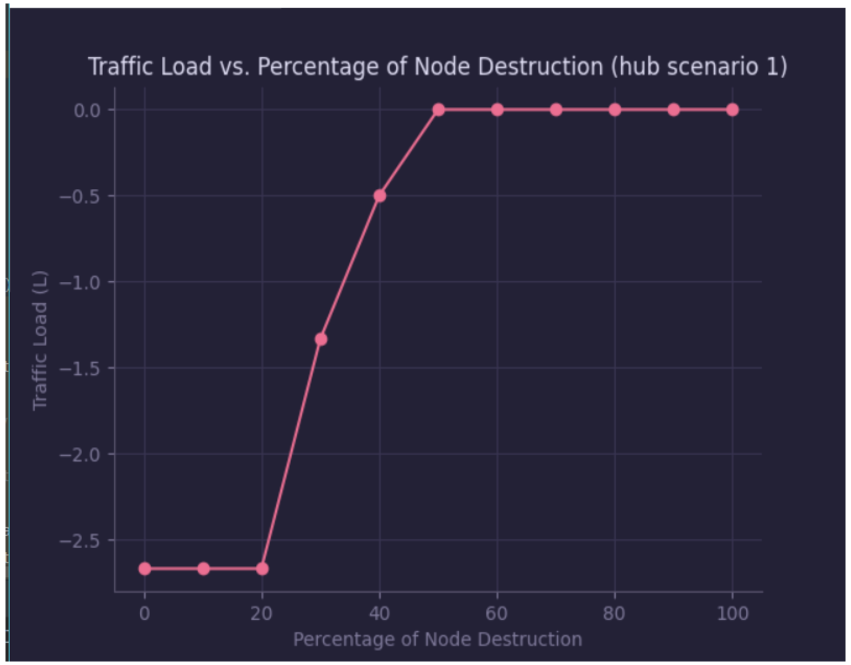
1.This will effect L and L max and make them to be greater.

2.The load balance is harder to be greater

3. the maximum number of edge-disjoint paths between two nodes in a graph is equal to the minimum number of edges that need to be removed to disconnect the two nodes

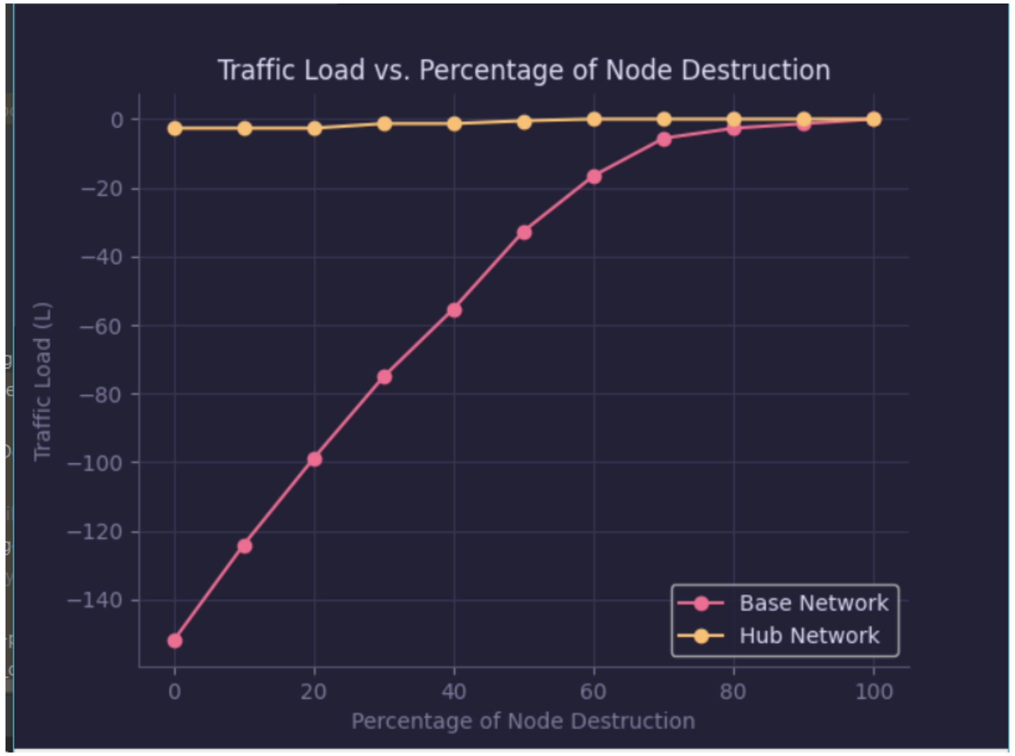
QUESTION7



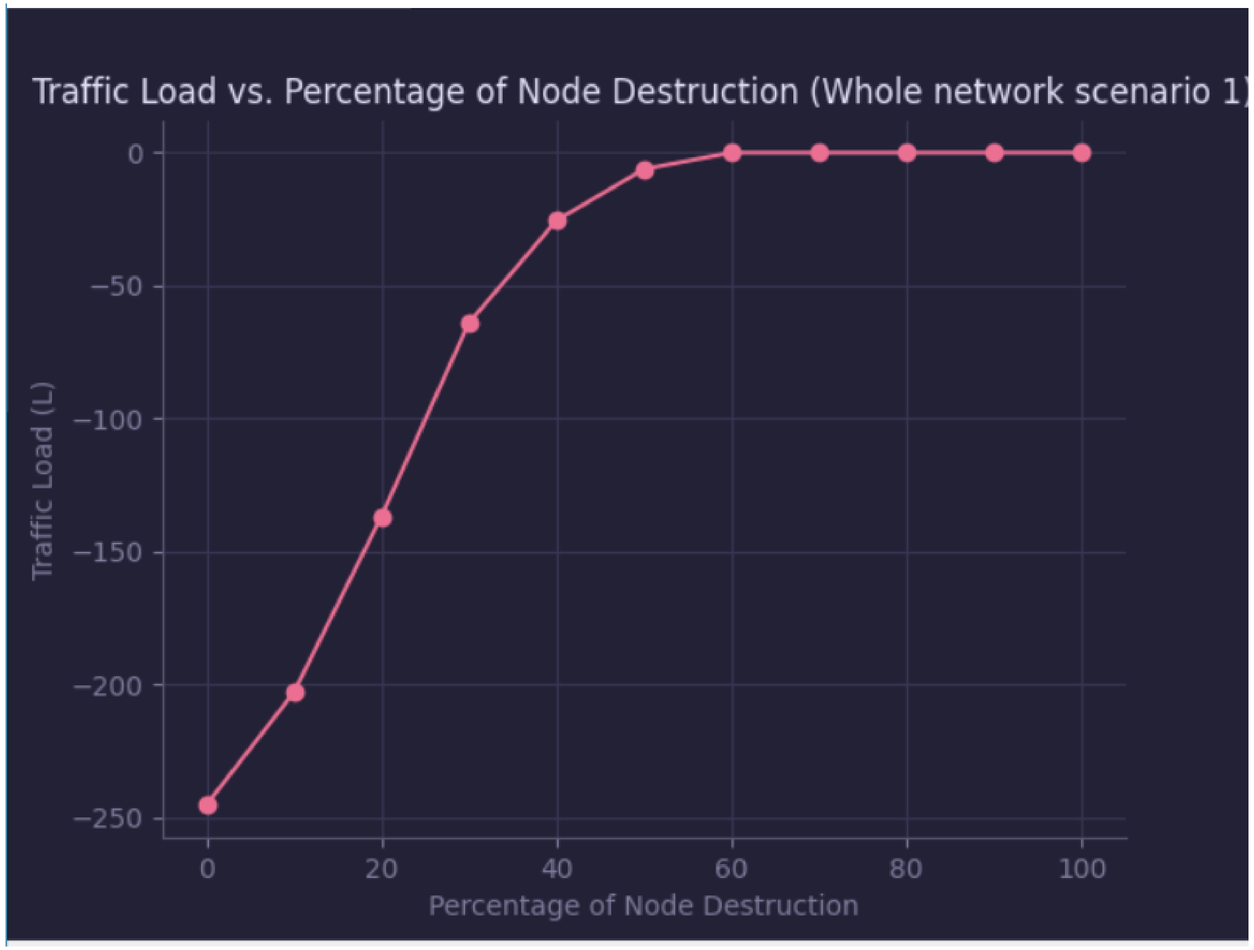


As it shown , 20 percentage hub is more better against destruction than base but the whole network has different story

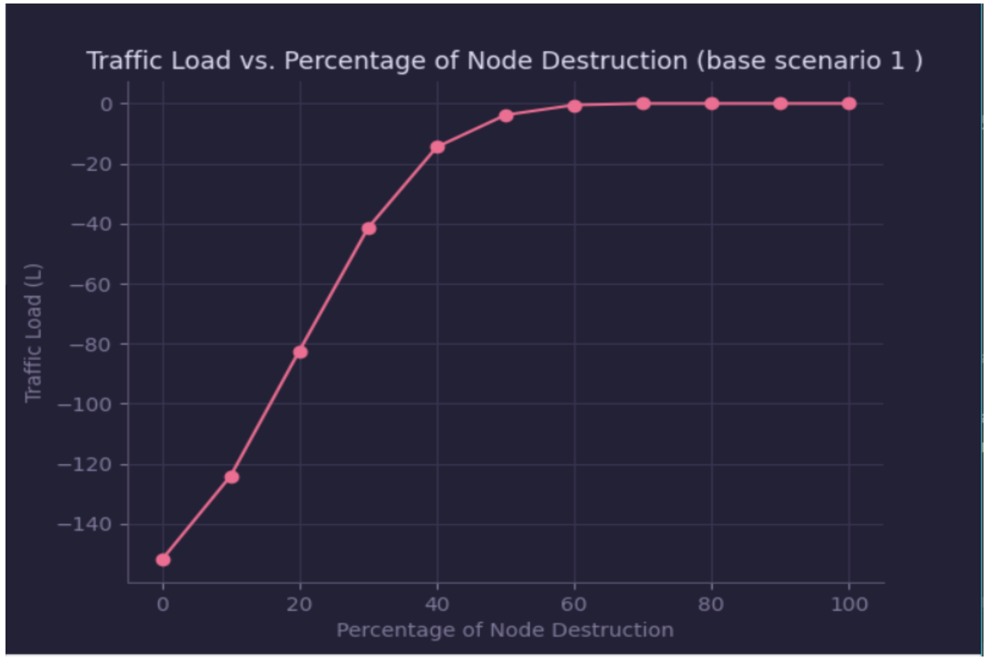
QUESTION7

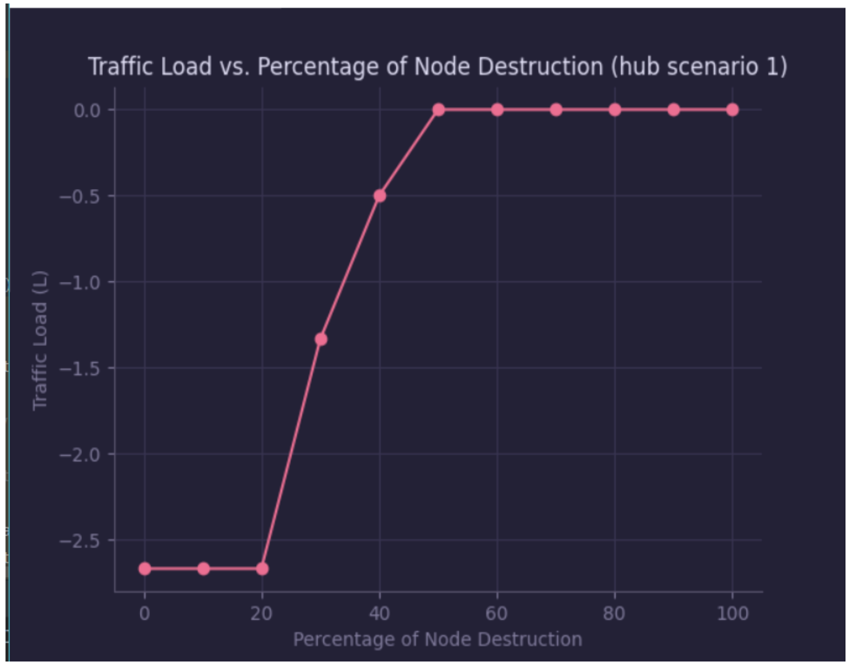


Here hub can ruin the network much faster than base



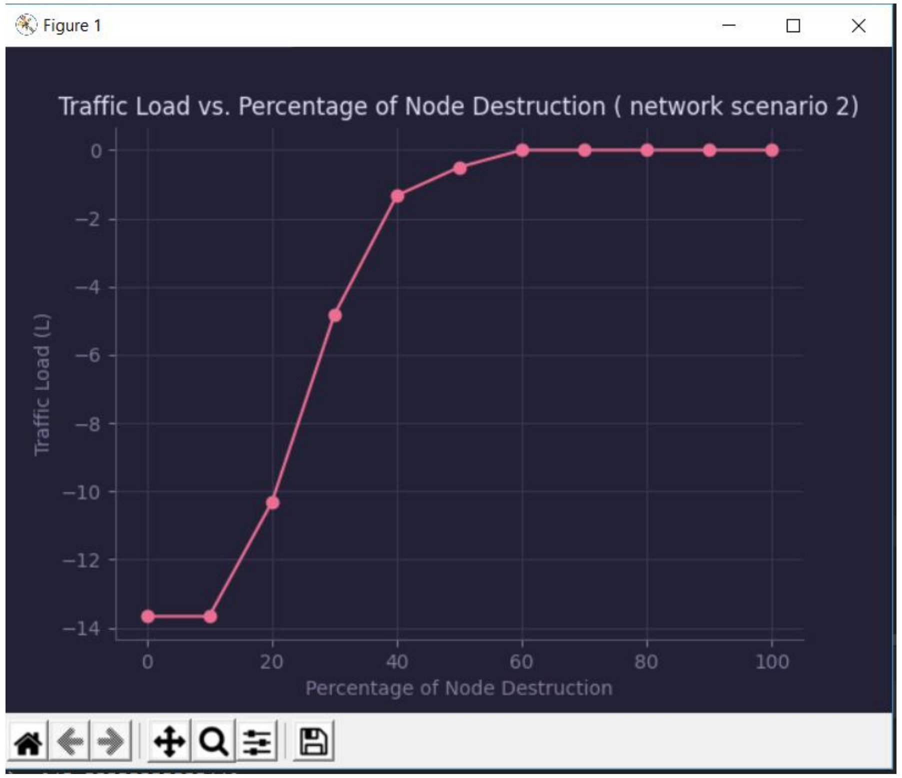
QUESTION7

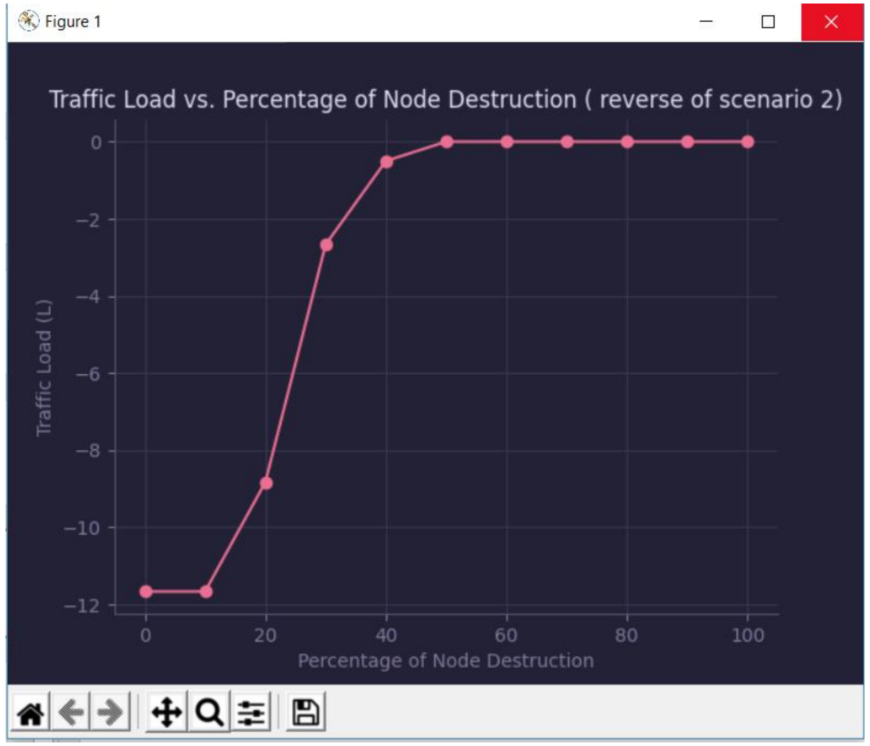




As it shown , 20 percentage hub is more better against destruction than base

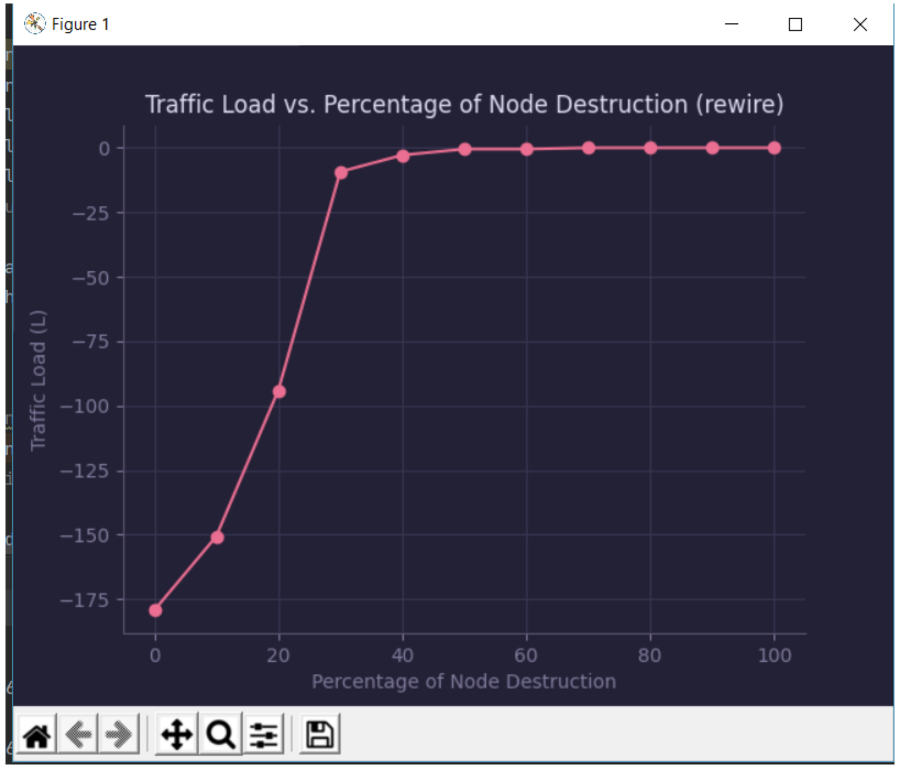
QUESTION7

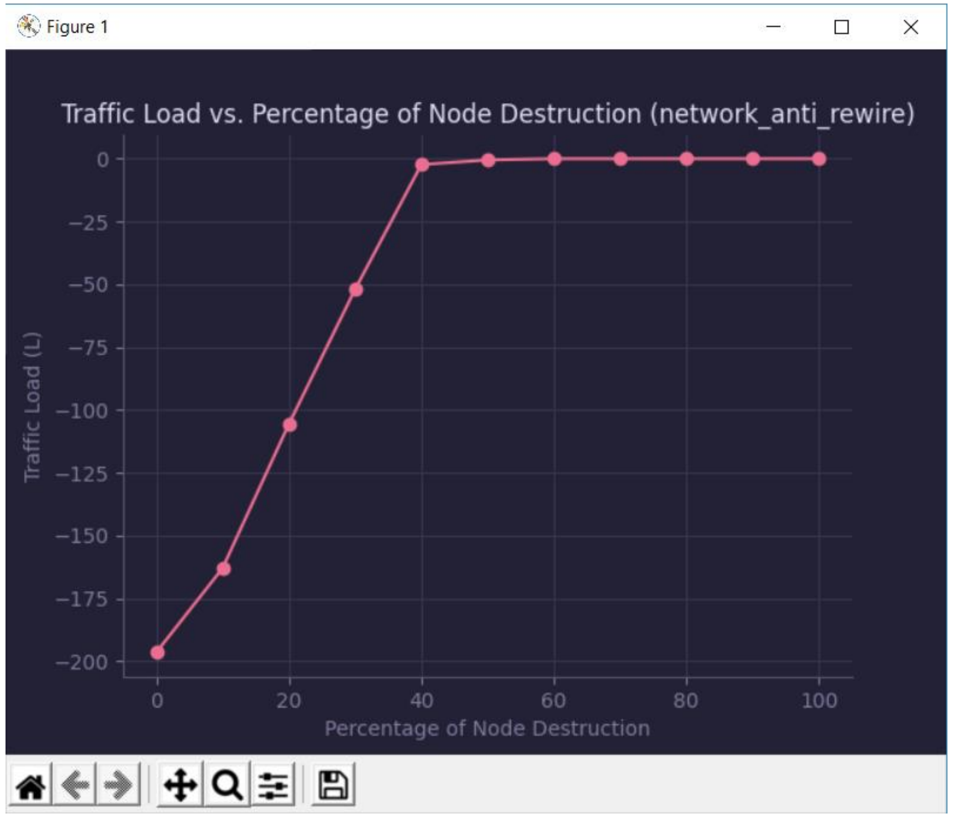




20 percentage different in reaching max traffic load , the first strategy is better

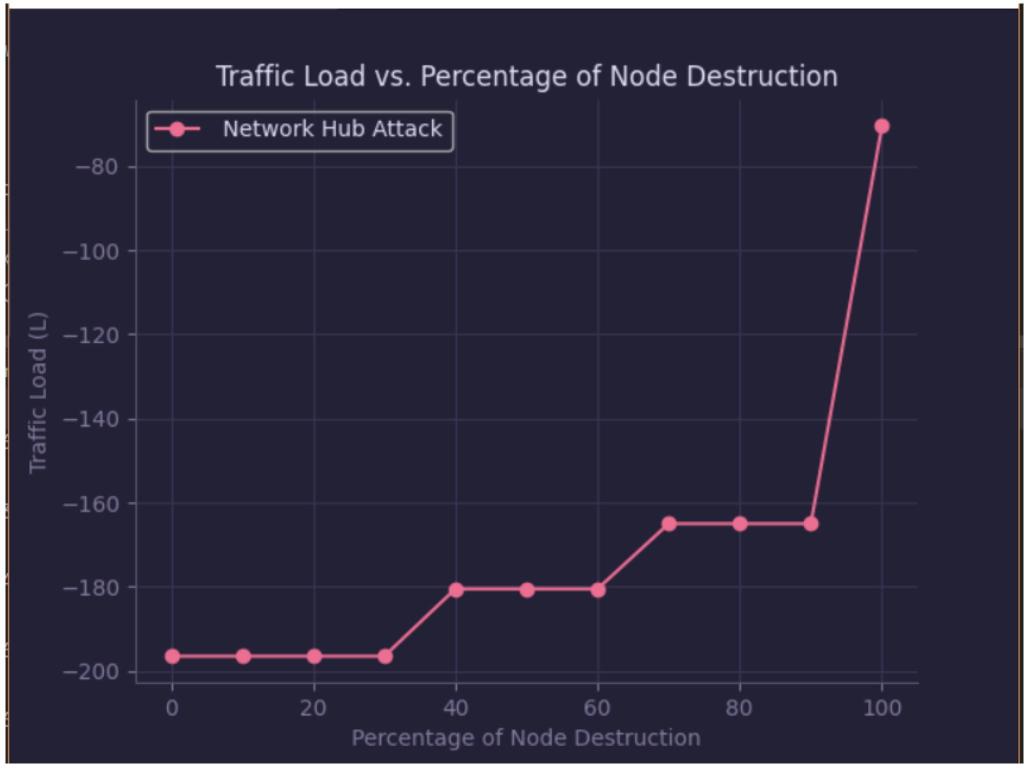
QUESTION7

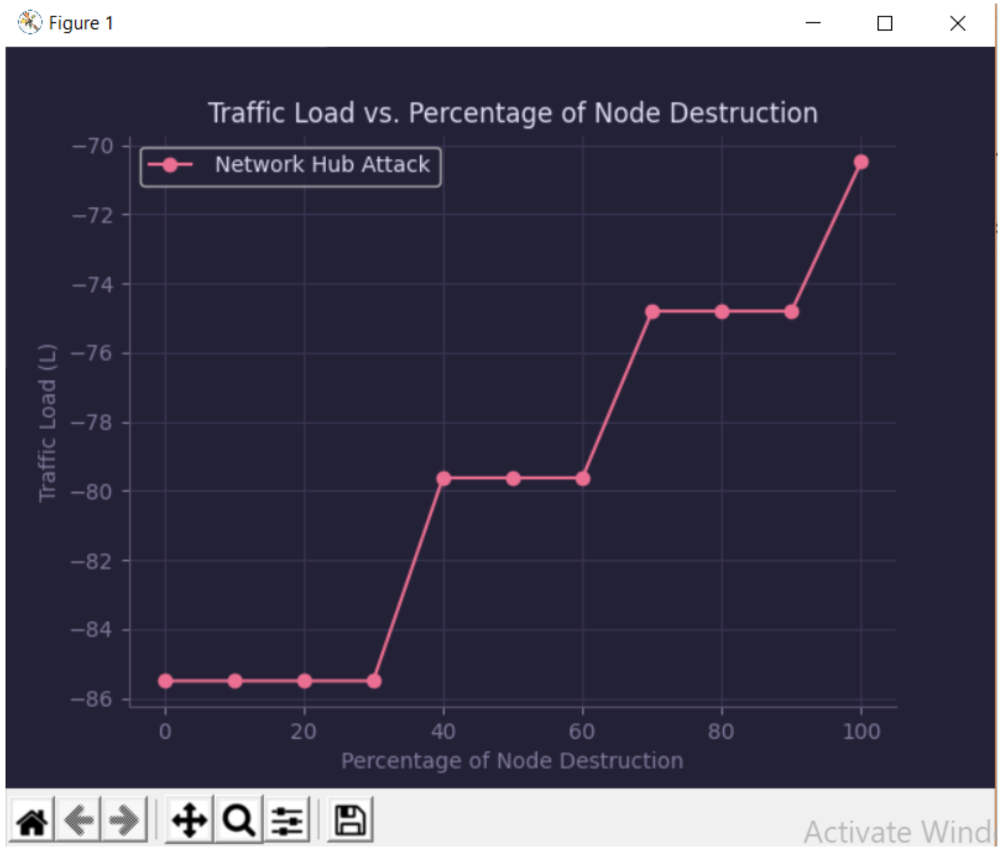




20 percentage different in reaching max traffic load , when changing wiring strategy

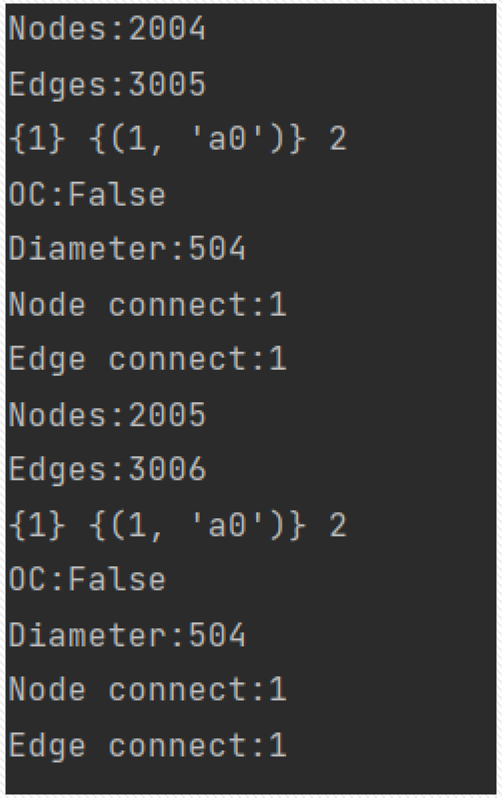
QUESTION7





No.1 is stronger against attack compared to No.2

QUESTION 8



```
Nodes:2004
Edges:3005
{1} {(1, 'a0')} 2
OC:False
Diameter:504
Node connect:1
Edge connect:1
Nodes:2005
Edges:3006
{1} {(1, 'a0')} 2
OC:False
Diameter:504
Node connect:1
Edge connect:1
```

Two wiring strategy implemented

One is semi-small world not always OC

Another is semi-BA   not always OC