Controlling Networks

By Kai McConnell

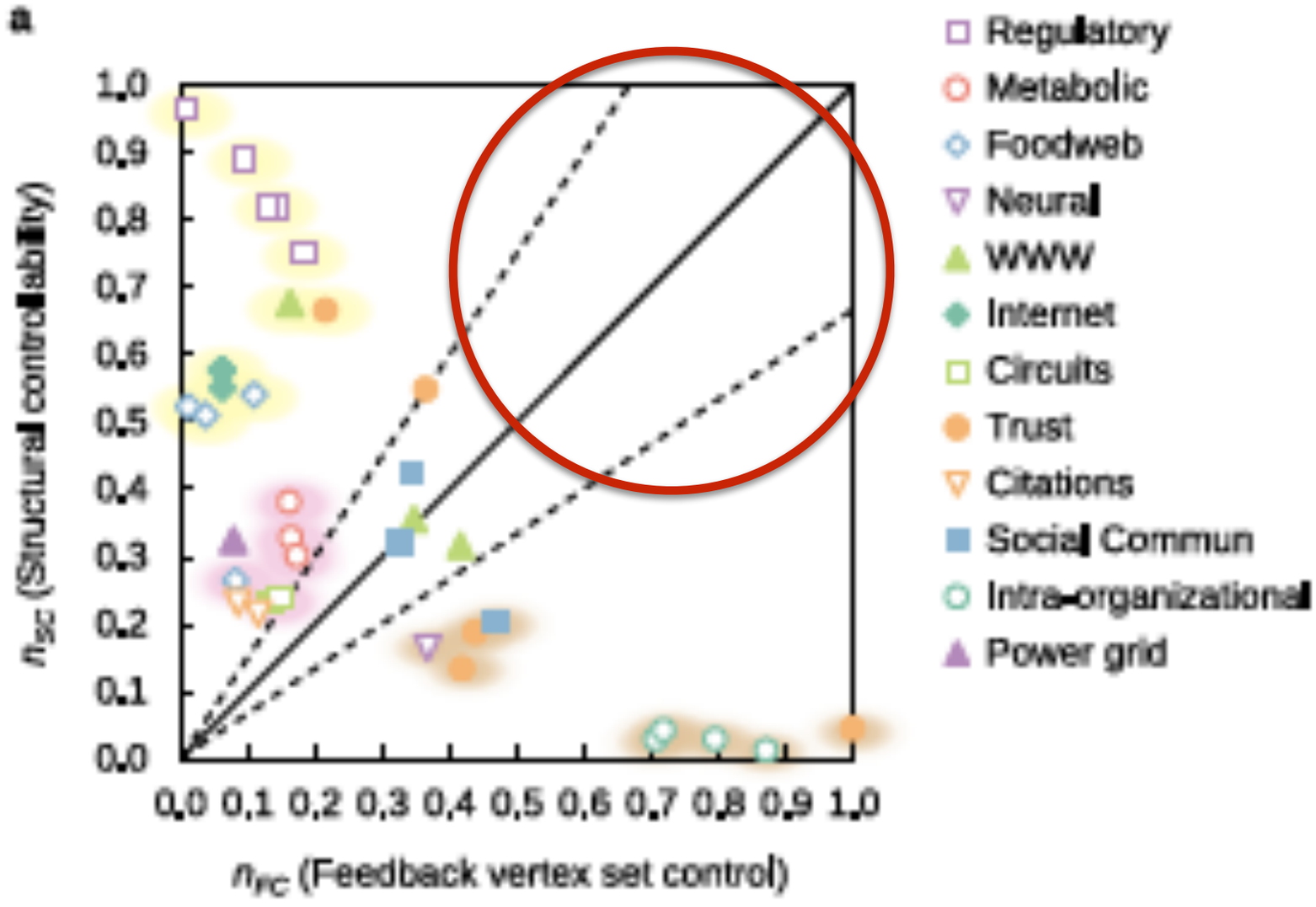
Introduction

- Minimal Feedback Vertex Set Control

- Structural Controllability

- Real Networks

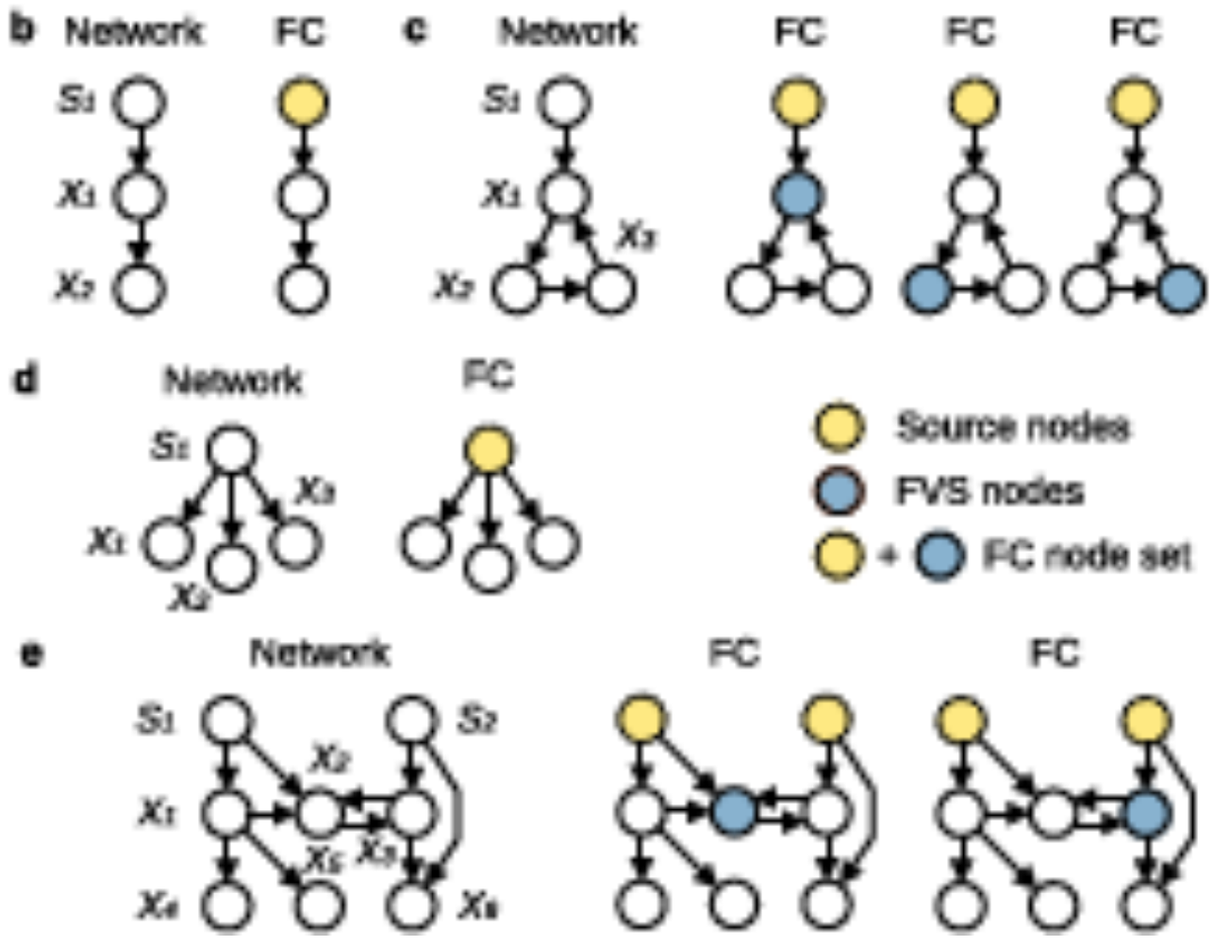
Goal:



Structure-based control of complex networks with nonlinear dynamics
by Jorge Gomez Tejeda Zañudo, Gang Yang, and Réka Albert

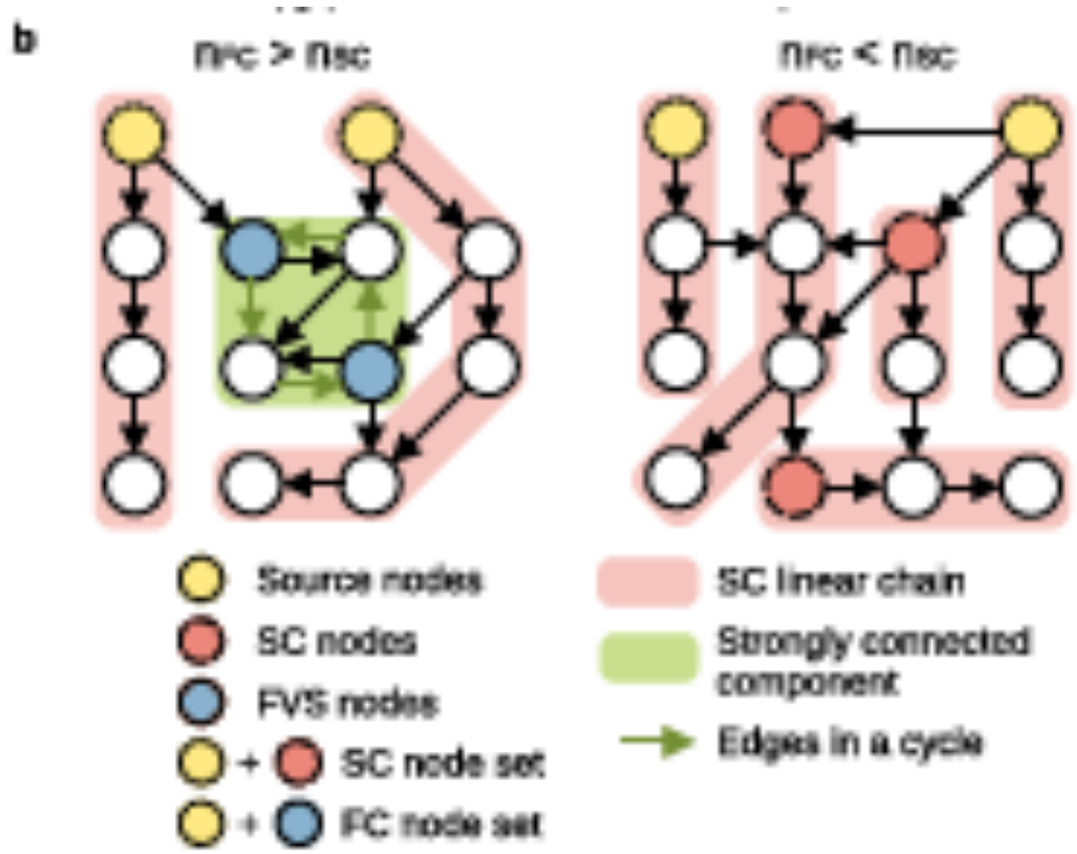
Minimal Feedback Vertex Set Control

- FC = |FVS| + |Source Nodes| / |V|
  - FC = Minimal Feedback Vertex Set Control
  - |FVS| = Number of nodes minimum required to create an acyclic graph
  - |Source Nodes| = Nodes with no incoming edges
  - V = Number of nodes



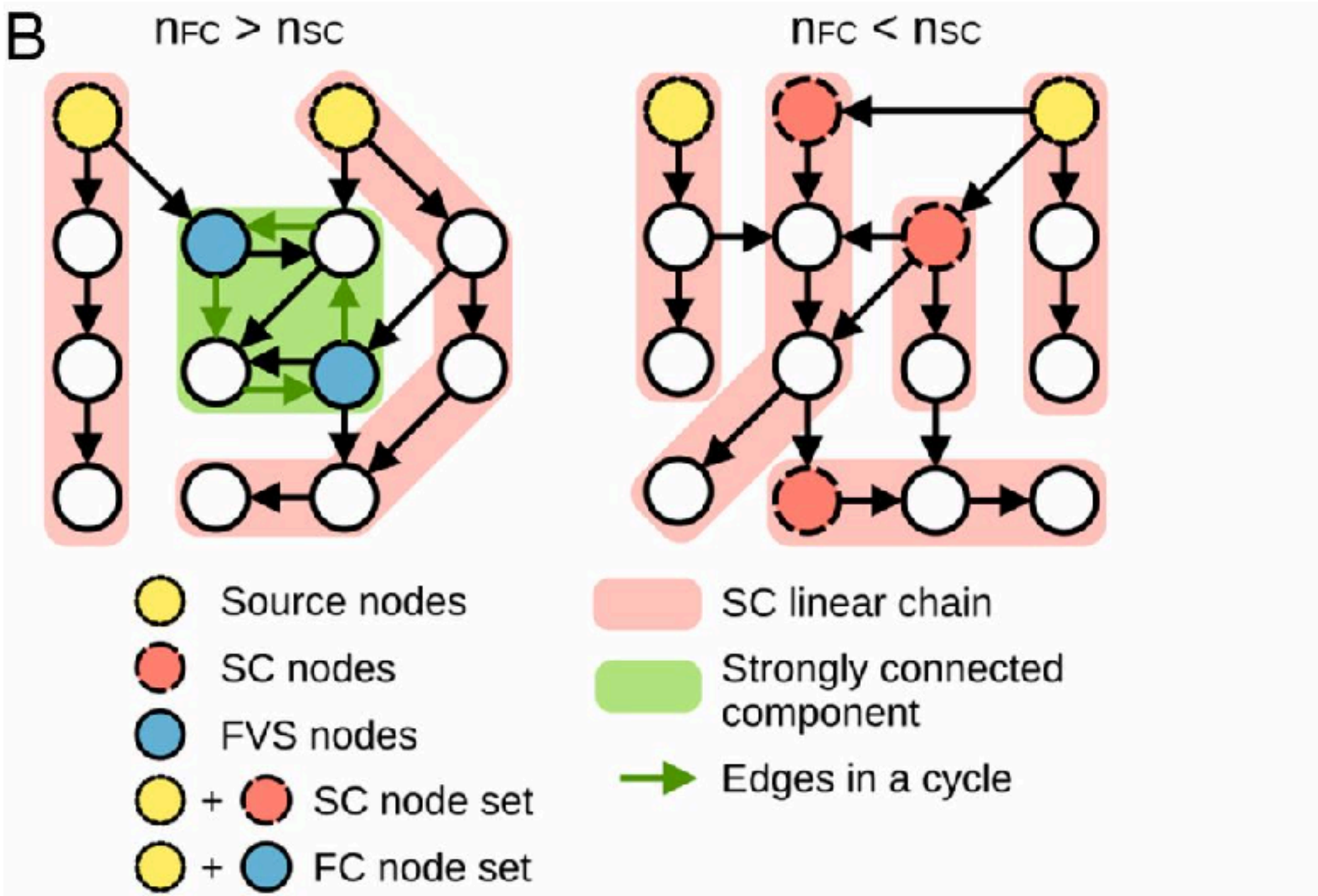
Structural Controllability

- SC = |Driver Nodes| / |V|
  - SC = Structural Controllability
  - |V| = number of nodes
  - |Driver Nodes| = number of nodes with no incoming edges as a result of bipartite matching



Structure-based control of complex networks with nonlinear dynamics
by Jorge Gomez Tejeda Zañudo, Gang Yang, and Réka Albert

Whats the difference?



Structure-based control of complex networks with nonlinear dynamics
by Jorge Gomez Tejeda Zañudo, Gang Yang, and Réka Albert

My work

- Update Yuhao's work to be usable with networks 2.0
    - Calculate SC
- Calculate FC
- Figure out trends in aiming for high of both values
- Convert and Explore real world networks

Calculate SC

```
for a in graph.nodes():
        newGraphA.add_node(str(a) + "+")
    newGraphB = nx.Graph()
    for b in graph.nodes():
        newGraphB.add_node(str(b) + "-")
    undirected = nx.Graph()

undirected.add_nodes_from(newGraphA.n
odes(), bipartite=0)

undirected.add_nodes_from(newGraphB.n
odes(), bipartite=1)
    for src, dst in graph.edges():
        newSrc = str(src) + "+"
        newDst = str(dst) + "-"
        undirected.add_edge(newSrc,
newDst)
    halfList = []
    for node in undirected.nodes():
        if node.endswith("-"):
            halfList.append(node)
matching =
nx.algorithms.bipartite.matching.hopcro
ft_karp_matching(undirected, halfList)
    unmatched = len(graph.nodes()) -
(len(matching) / 2)
```

```
G = nx.read_edgelist("toy_copy.txt",
create_using=nx.DiGraph())

G_undirected = nx.Graph()
G_undirected.add_nodes_from(G.nodes())
duplicate_nodes = [-1 * node for node in
G.nodes()]
G_undirected.add_nodes_from(duplicate_
nodes)
for e in G.edges():
    source, target = e
    G_undirected.add_edge(source, -1 *
target)
G_undirected.edges()
matching =
nx.algorithms.bipartite.matching.hopcro
ft_karp_matching(G_undirected)
unmatched = len(G.nodes()) -
len(matching) / 2
G_source = []
for node in G.nodes():
    if G.in_degree(node) == 0:
        G_source.append(node)
control = (unmatched + len(G_source)) /
len(G.nodes())
print(control)
```

Calculate FC

```
#Just try all combinations of nodes in the graph, for each combination check if the induced
subgraph is acyclic#

def get_fbvs(graph):
    if is_acyclic(graph):
        return set()
# remove all nodes of degree 0 or 1 as they can't be part of any cycles
remove_node_deg_01(graph)

nodes = graph.nodes()
for L in range(0, len(nodes) + 1):
    for subset in itertools.combinations(nodes, L):
        # make an induced subgraph with the current node subset removed
        new_graph = graph_minus(graph, subset)

        if is_acyclic(new_graph):
            return subset

return set()
```

Converting Real World Networks

- Used Cytoscape to create tables

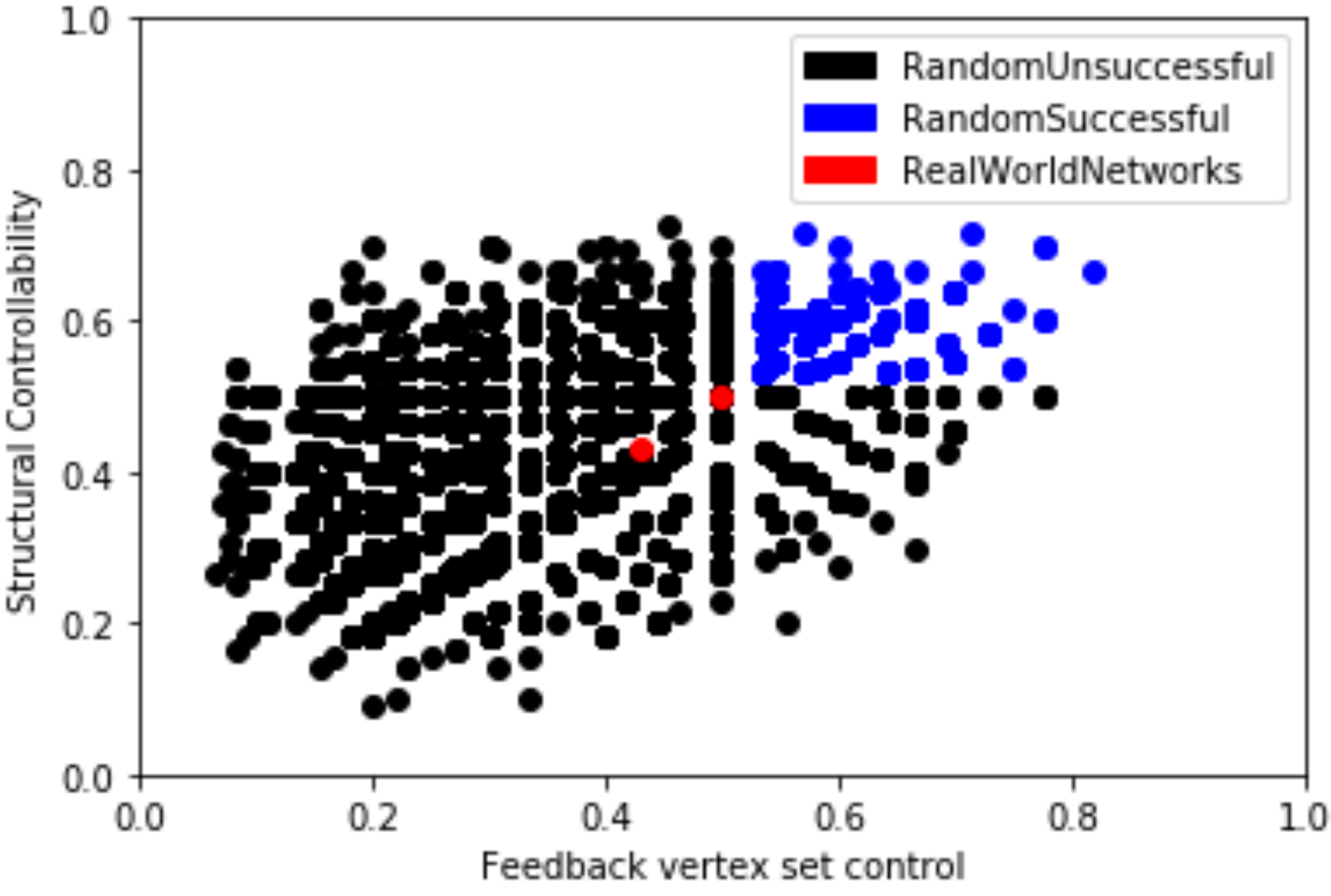
- Used numbers to remove unwanted columns and rows

- Used pages and copy pasting to convert table to tab separated txt documents

Results

- Increasing cutoff for what constitutes a success also lowers likelihood of success when it comes to high FC and SC

Success Distribution



Relationship between Nodes, Edges, and Success

- If there are less edges than nodes it is highly likely to find high of both FC and SC (Because parts of the graph are disconnected)

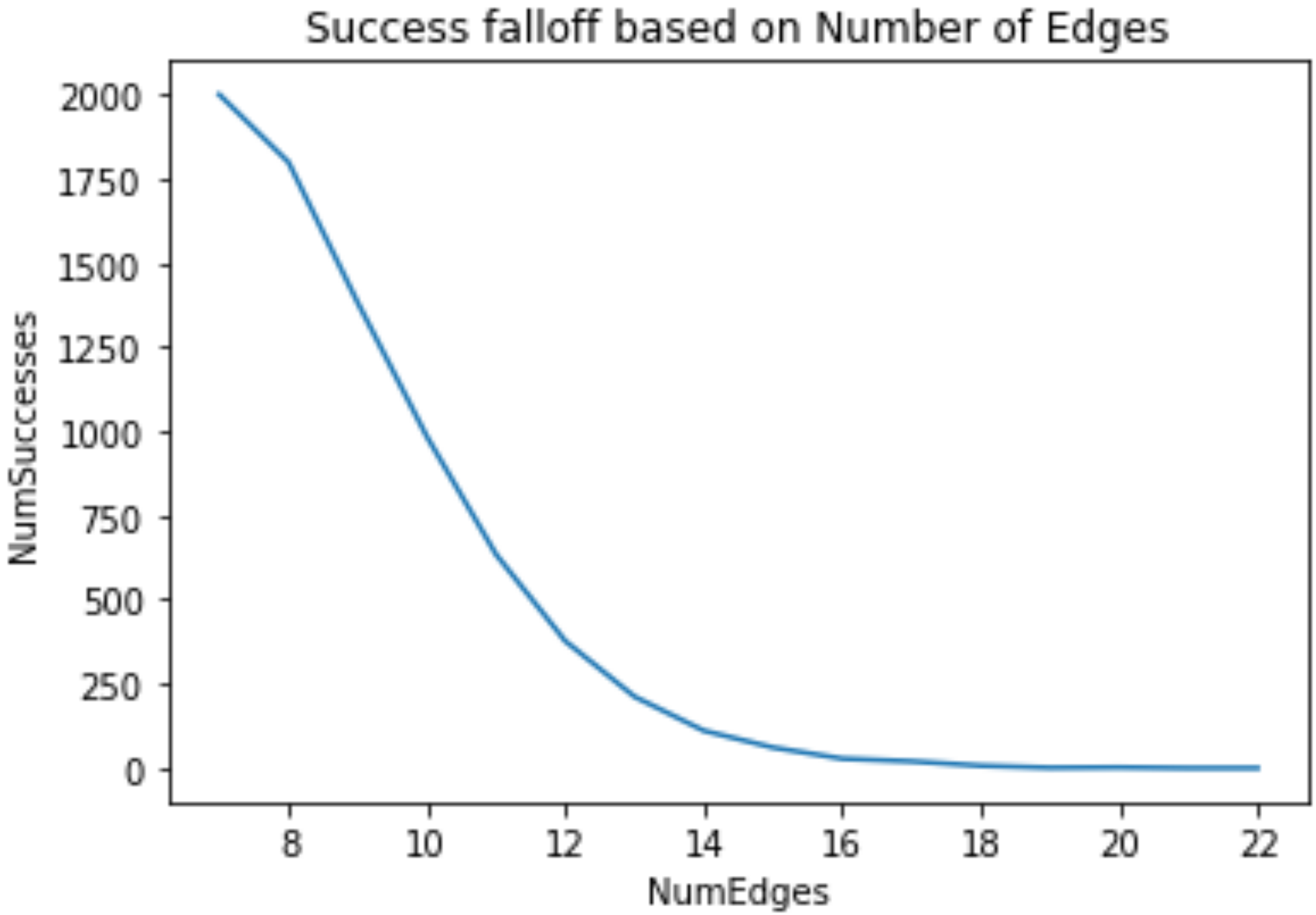
- If there are more edges than nodes it is unlikely to find high of both FC and SC

- Successes Guaranteed if number of edges is equal to half the number of nodes

- Success Impossible if number of edges is greater than 2 times the number of nodes

    - Ran test 20000 times on graph with 15 nodes and 30 edges and received no successful results

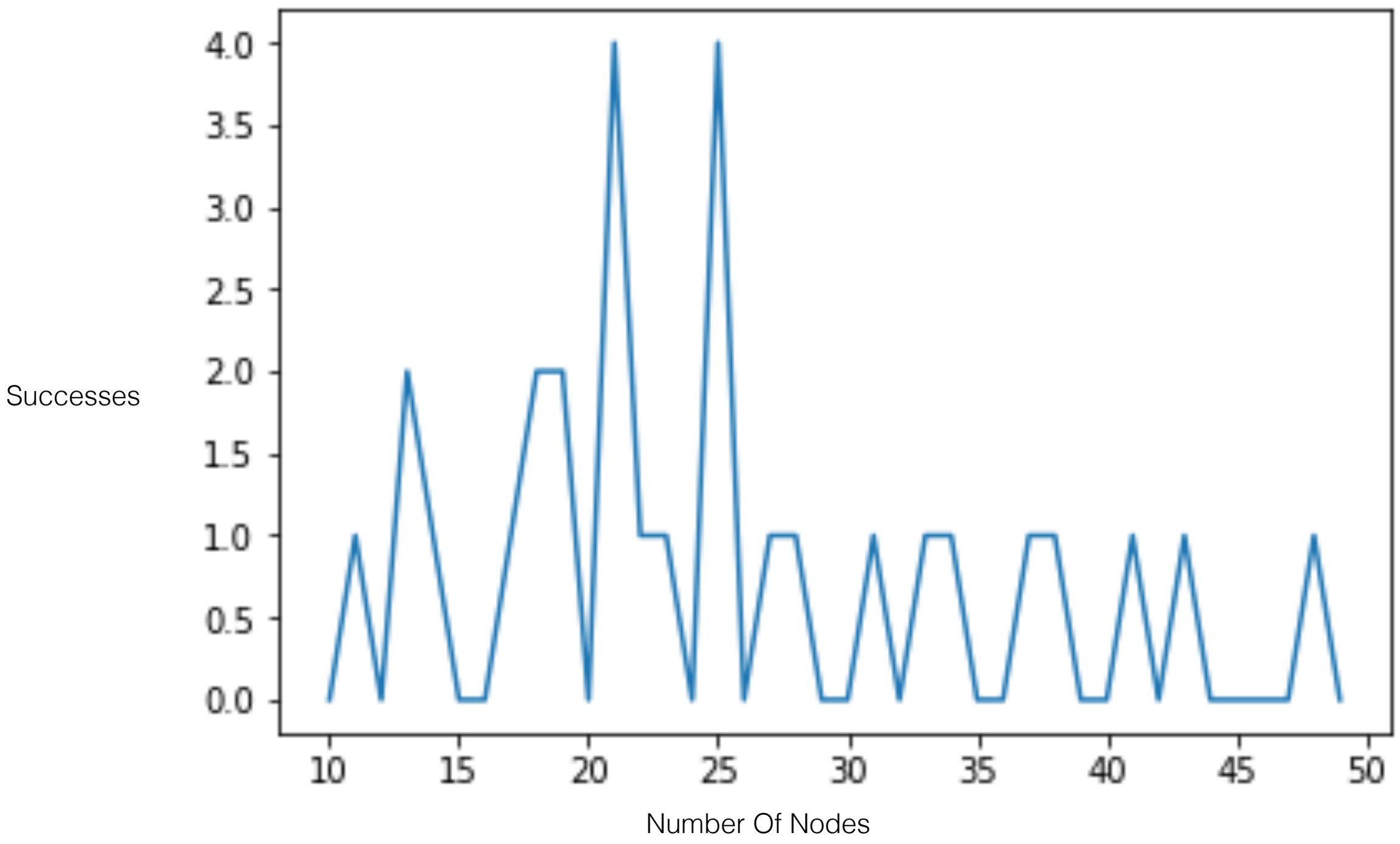
1000 Iterations Up to 100 Edges on 10 Nodes



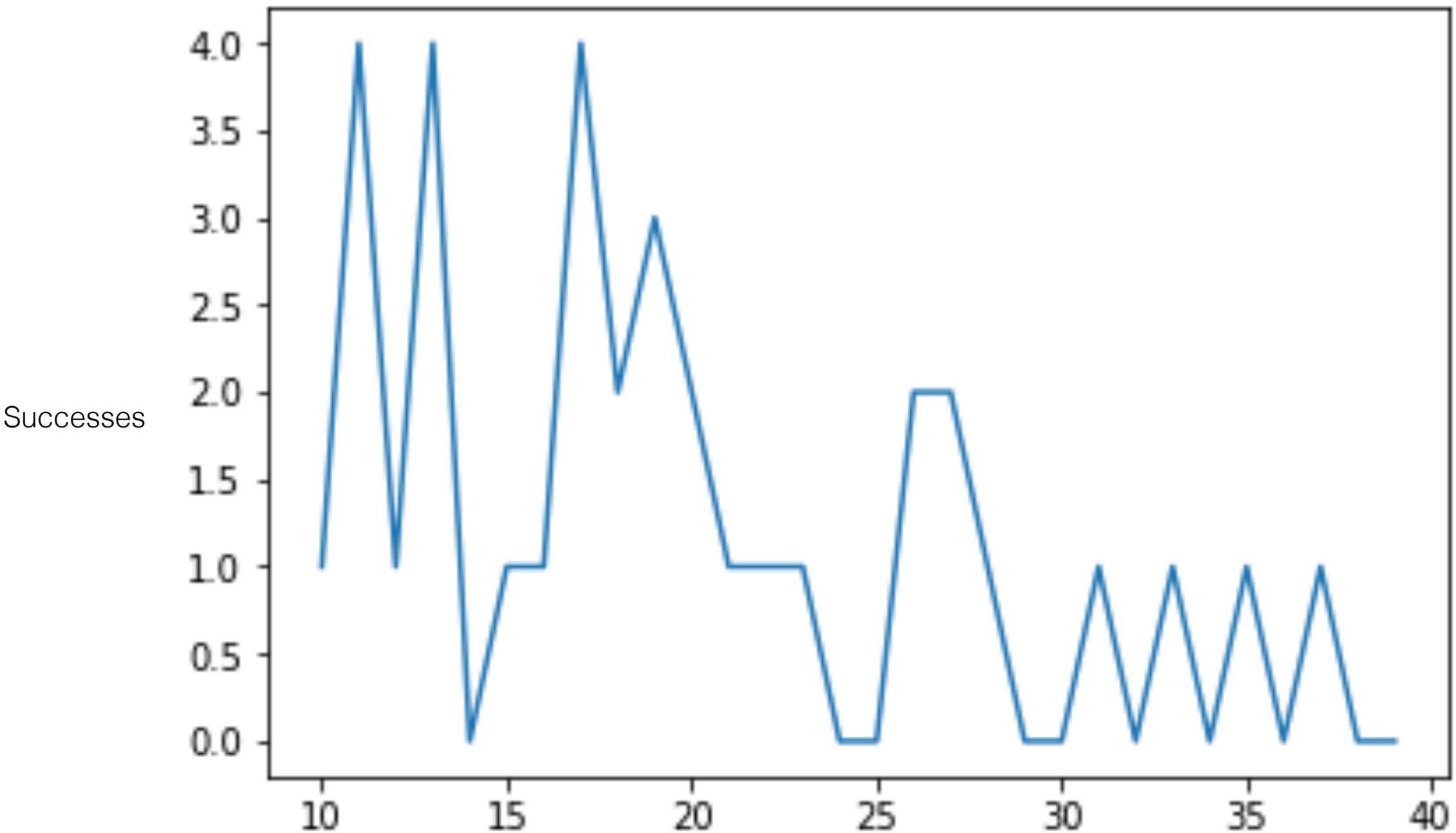
Relationship between Nodes and High FC and SC

- The higher number of nodes the less likely a graph will result in high FC and SC

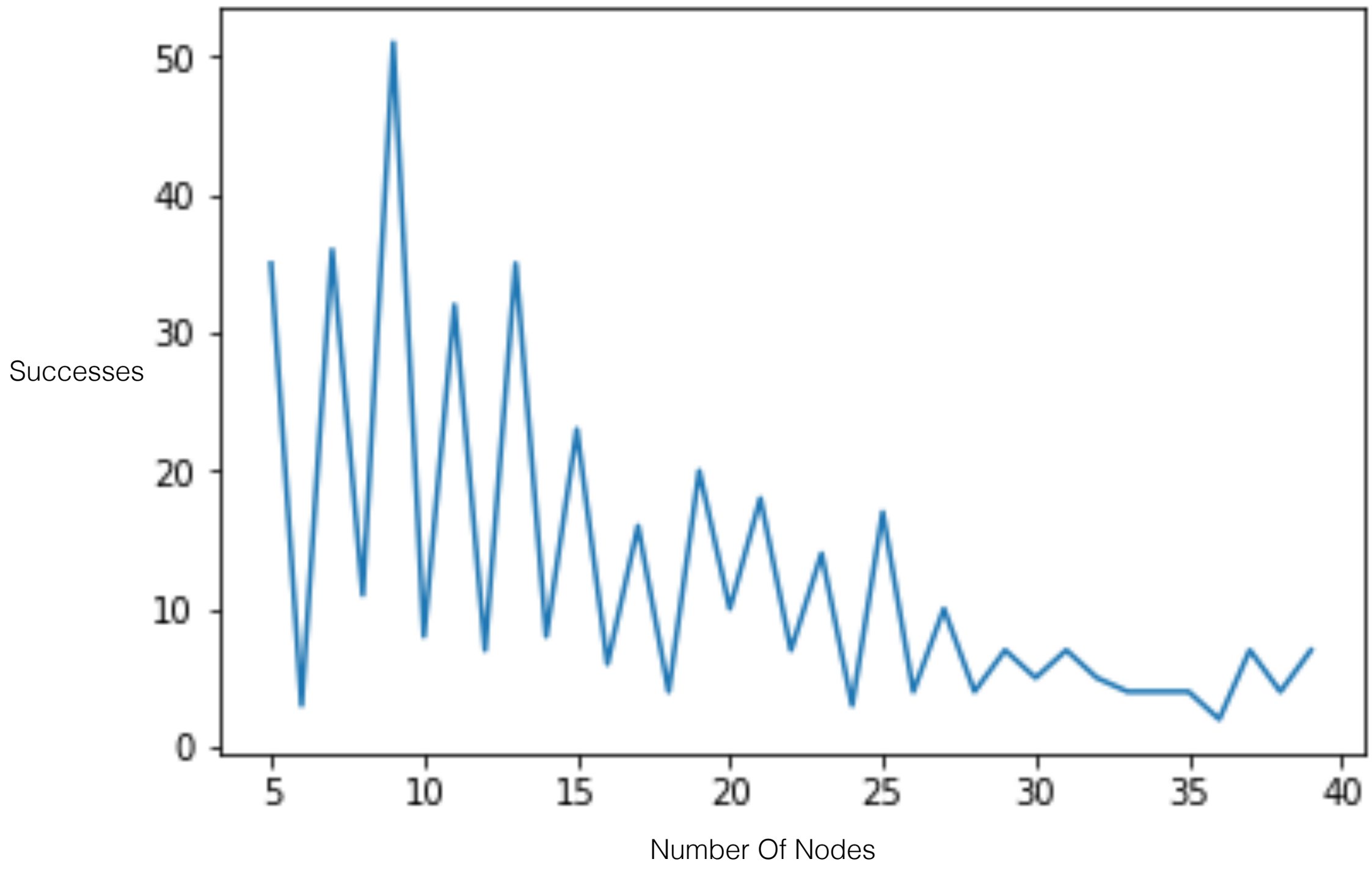
50 iterations 50 nodes



100 Iterations, 40 Nodes



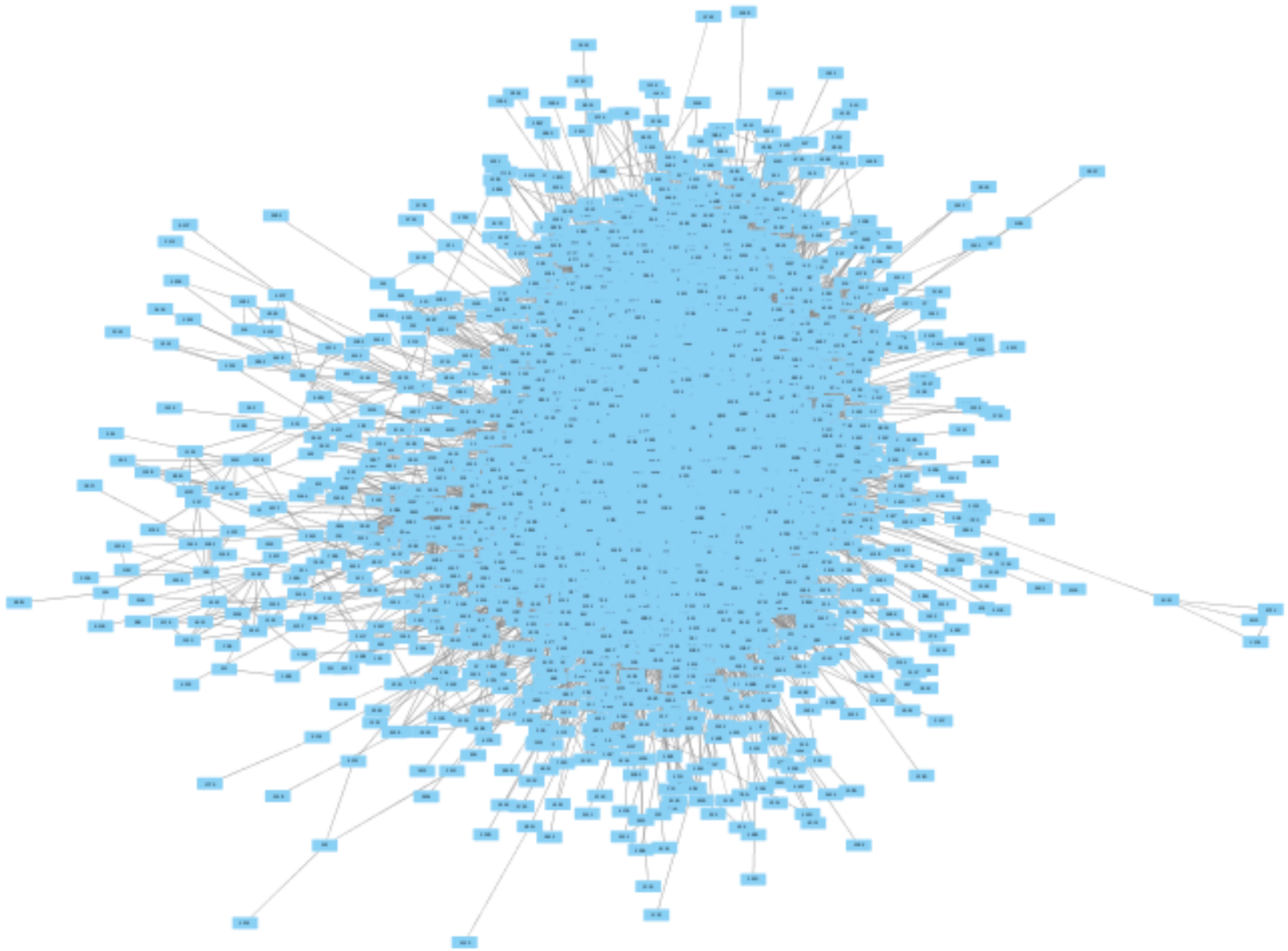
1000 Iterations, 40 Nodes



Real World graphs

- Best Example I found was facebook Caltech social network
    - FC = 0.5
    - SC = 0.5
- Other Examples Include:
    - Airline Travel
        - FC = 0.429553264604810 97
        - SC = 0.424583244604713 92
    - Etc.

Airport Traffic Graph



http://networkrepository.com/

So What?

- Unlikely for real world networks to have high SC and FC because of large numbers of nodes and edges
    - This can become more likely for specific kind of real world networks under such as facebook social networks
- It becomes harder and harder to control a network as it gets larger

Extensions:

- Better Algorithm for calculating the two values (Ran into problems with large networks)
- Analyze using different random network generators
- Do the analysis more and on larger randomly generated networks to get a better curve.
- Do many more real world Networks (Huge amount of computing Time)

References

- Csaba Kerepesi

- Yahoo Wan

- Structure-based control of complex networks with nonlinear dynamics by Jorge Gomez Tejeda Zañudo, Gang Yang, and Réka Albert

- https://snap.stanford.edu/data/

- http://networkrepository.com/