Python for Complex Network Analysis

Sajeev G P

February 17, 2020

Connected Graphs

An undirected graph is **connected** if, for every pair nodes, there is a path between them.

In: nx.is_connected(G)

Out: True

However, if we remove edges A—G, A—N, and J—O, the graph becomes disconnected.

Connected Graphs

An undirected graph is **connected** if, for every pair nodes, there is a path between them.

In: nx.is_connected(G)

Out: True

However, if we remove edges A—G, A—N, and J—O, the graph becomes disconnected.

There is no path between nodes in the three different "communities".

Graph Components

Connected component:
A subset of nodes such as:

i. Every node in the subset has a path to every other node.
ii. No other node has a path to any node in the subset.

Is the subset {E, A, G, F} a connected component?

No, there is no path between nodes A and F.

Graph Components

Connected component:
A subset of nodes such as:

i. Every node in the subset has a path to every other node.
ii. No other node has a path to any node in the subset.

Is the subset {N, O, K} a connected component?

No, node L has a path to N, O, and K.

Graph Components

Connected component:
A subset of nodes such as:

i. Every node in the subset has a path to every other node.
ii. No other node has a path to any node in the subset.

What are the connected components in this graph?

{A, B, C, D, E}, {F, G, H, I, J}, {K, L, M, N, O}

Graph Components

```
In: nx.number_connected_components(G)
Out: 3

In: sorted(nx.connected_components(G))
Out: [{'A', 'B', 'C', 'D', 'E'},
{'F', 'G', 'H', 'I', 'J'},
{'K', 'L', 'M', 'N', 'O'}]

In: nx.node_connected_component(G, 'M')
Out: {'K', 'L', 'M', 'N', 'O'}
```

Connectivity in Directed Graphs

A directed graph is **strongly connected** if, for every pair nodes *u* and *v*, there is a directed path from u to v and a directed path from *v* to *u*.

In: nx.is_strongly_connected(G)
Out: False

Note: There is no directed path from A to H

Connectivity in Directed Graphs

A directed graph is **weakly connected** if replacing all directed edges with undirected edges produces a connected undirected graph.

Connectivity in Directed Graphs

A directed graph is **weakly connected** if replacing all directed edges with undirected edges produces a connected undirected graph.

In: nx.is_weakly_connected(G)
Out: True

Connectivity in Directed Graphs

Strongly connected component:
A subset of nodes such as:

i. Every node in the subset has a **directed** path to every other node.
ii. No other node has a **directed** path to every node in the subset.

What are the strongly connected components in this graph?

In: sorted(nx.strongly_connected_components(G))

Out: [{M}, {L}, {K}, {A, B, C, D, E, F, G, J, N, O}, {H, I}]

Connectivity in Directed Graphs

Weakly connected component:

The connected components of the graph after replacing all directed edges with undirected edges.

In: sorted(nx.weakly_connected_components(G))

Out: [{'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O'}]

Since the graph is weakly connected it only has one weakly connected component.

Summary

Undirected Graphs

Connected: for every pair nodes, there is a path between them.

Connected components

nx.connected_components(G)

Directed Graphs

Strongly connected: for every pair nodes, there is a *directed* path between them.

Strongly connected components

nx.strongly_connected_components(G))

Connectivity and Robustness in Networks

Network robustness: the ability of a network to maintain its general structural properties when it faces failures or attacks.

Type of attacks: removal of nodes or edges.

Structural properties: connectivity.

Examples: airport closures, internet router failures, power line failures.

Network of direct flights around the world [Bio.Diaspora]

Internet Connectivity [K. C. Claffy]

Disconnecting a Graph

What is the smallest number of nodes that can be removed from this graph in order to disconnect it?

```
In: nx.node_connectivity(G_un)
Out: 1
```

Which node?

```
In: nx.minimum_node_cut(G_un)
Out: {'A'}
```

Disconnecting a Graph

What is the smallest number of nodes that can be removed from this graph in order to disconnect it?

In: nx.node_connectivity(G_un)
Out: 1

Which node?
In: nx.minimum_node_cut(G_un)
Out: {'A'}

Disconnecting a Graph

What is the smallest number of **nodes** that can be removed from this graph in order to disconnect it?

```
In: nx.node_connectivity(G_un)
Out: 1
```

Which node?

```
In: nx.minimum_node_cut(G_un)
Out: {'A'}
```

Disconnecting a Graph

What is the smallest number of **edges** that can be removed from this graph in order to disconnect it?

```
In: nx.edge_connectivity(G_un)
Out: 2
```

Which edges?

```
In: nx.minimum_edge_cut(G_un)
Out: {('A', 'G'), ('O', 'J')}
```

Disconnecting a Graph

What is the smallest number of **edges** that can be removed from this graph in order to disconnect it?

In: nx.edge_connectivity(G_un)
Out: 2

Which edges?
In: nx.minimum_edge_cut(G_un)
Out: {('A', 'G'), ('O', 'J')}

Disconnecting a Graph

What is the smallest number of **edges** that can be removed from this graph in order to disconnect it?

In: nx.edge_connectivity(G_un)
Out: 2

Which edges?
In: nx.minimum_edge_cut(G_un)
Out: {('A', 'G'), ('O', 'J')}

Robust networks have large minimum node and edge cuts.

Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

```
In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out: [['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]
```

Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out: [**['G', 'A', 'N', 'L']**,
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]

Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out: [['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]

Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

```
In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out: [['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]
```

Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out: [['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]

Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out: [['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]

Node Connectivity

If we wanted to block the message from G to L by removing **nodes** from the network, how many nodes would we need to remove?

In: nx.node_connectivity(G, 'G', 'L')

Out: 2

Which nodes?

In: nx.minimum_node_cut(G, 'G', 'L')

Out: {'N', 'O'}

Node Connectivity

If we wanted to block the message from G to L by removing **nodes** from the network, how many nodes would we need to remove?

In: nx.node_connectivity(G, 'G', 'L')

Out: 2

Which nodes?

In: nx.minimum_node_cut(G, 'G', 'L')

Out: {'N', 'O'}

If we only remove node N, message can go on path G → J → O → K → L.

Node Connectivity

If we wanted to block the message from G to L by removing **nodes** from the network, how many nodes would we need to remove?

In: nx.node_connectivity(G, 'G', 'L')

Out: 2

Which nodes?

In: nx.minimum_node_cut(G, 'G', 'L')

Out: {'N', 'O'}

If we only remove node O, message can go on path G → A → N → L.

Edge Connectivity

If we wanted to block the message from G to L by removing **edges** from the network, how many edges would we need to remove?

In: nx.edge_connectivity(G, 'G', 'L')

Out: 2

Which edges?

In: nx.minimum_edge_cut(G, 'G', 'L')

Out: {('A', 'N'), ('J', 'O')}

Edge Connectivity

If we wanted to block the message from G to L by removing **edges** from the network, how many edges would we need to remove?

In: nx.edge_connectivity(G, 'G', 'L')

Out: 2

Which edges?

In: nx.minimum_edge_cut(G, 'G', 'L')

Out: {('A', 'N'), ('J', 'O')}

We need to remove A → N an J → O to block messages from G to L.

Summary

Node connectivity: Minimum number of *nodes* needed to disconnect a graph or pair of nodes.

nx.node_connectivity(G, 'G', 'L')
nx.minimum_node_cut(G, 'G', 'L')

Edge connectivity: Minimum number of *edges* needed to disconnect a graph or pair of nodes.

nx.edge_connectivity(G, 'G', 'L')
nx.minimum_edge_cut(G, 'G', 'L')

Graphs with large node and edge connectivity are more robust to the loss of nodes and edges.

Resources

- NetworkX Docs
  `http://networkx.lanl.gov/tutorial/index.html`
- NetworkX Tutorial
  `http://networkx.lanl.gov/contents.html`
- Matplotlib Docs
  `http://matplotlib.sourceforge.net/contents.html`
- Matplotlib Tutorial
  `http://matplotlib.sourceforge.net/users/pyplot_tutorial.html`
- Numpy Docs
  `http://numpy.scipy.org/`
- MacPorts
  `http://macports.org`