Network Visualization using NetworkX and Matplotlib

Jana Lasser

MPI for Dynamics and Self-Organization Göttingen

Outline

- Plotting Nodes/Edges
- Graph Layouts
- Colors
- Plotting with Edge/Node Attributes

Graph Plotting Basics

Drawing functions:

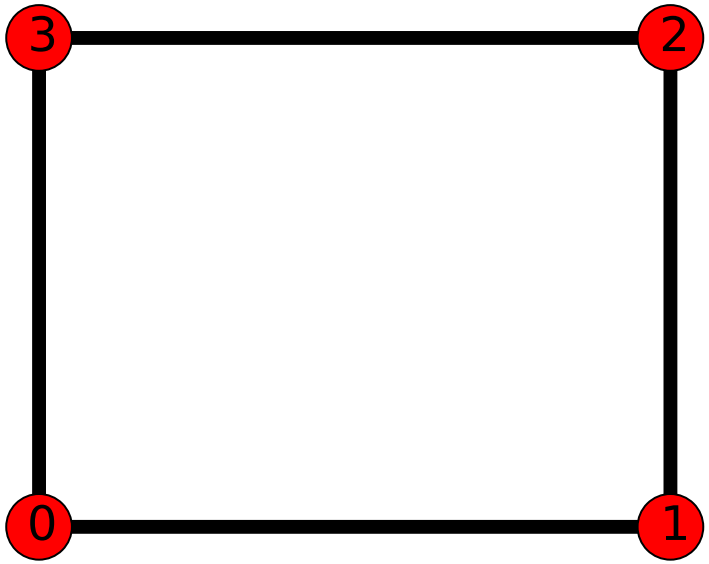
```
#draws edges as lines and nodes as dots
nx.draw_networkx(G,pos)
#draws only nodes
nx.draw_networkx_nodes(G,pos)
#draws only edges
nx.draw_networkx_edges(G,pos)
```

Input:
Graph `G` as before
Dictionary of node coordinates

```
pos = {0:[1.0,1.0], 1:[0.5,2.0], ...}
```

Example

```
G = nx.Graph()
G.add_nodes_from([0,1,2,3])
G.add_edges_from([(0,1),(1,2),(2,3),(3,0)])
pos = {0:[0,0], 1:[1,0], 2:[1,1], 3:[0,1]}
nx.draw_networkx(G,pos)
```



Graph Layouts

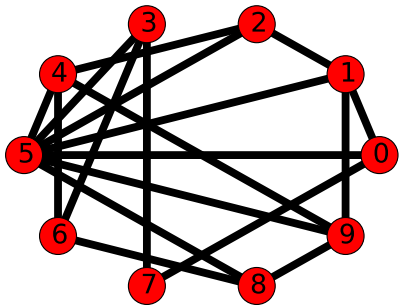
It is tedious to create the list of node positions by hand.
Often graphs don't even have a geometry to use ⇒ layouts.

- `circular_layout` - *positions nodes on circle*
- `random_layout` - *random but uniform positions in unit square*
- `shell_layout` - *positions nodes in concentric circles*
- `spring_layout` - *few crossings, edges of more or less equal length*
- `spectral_layout` - *uses eigenvectors of graph laplacian as coordinates*

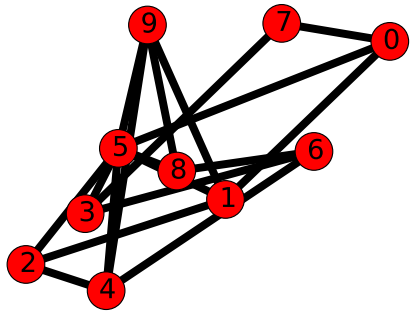
```
G = nx.gnp_random_graph(10,0.3)
pos = nx.circular_layout(G)
nx.draw_networkx(G,pos)
```

Layout Examples

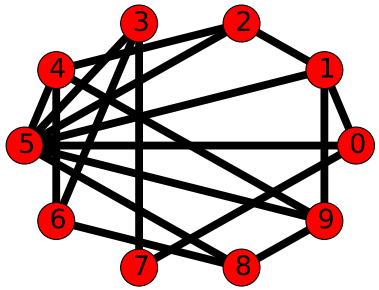
Circular



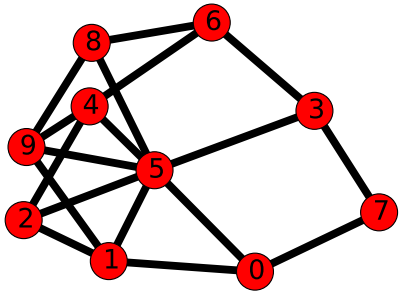
Random



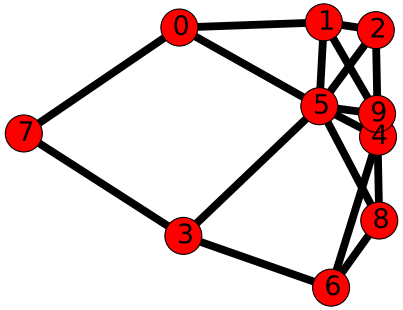
Shell



Spring



Spectral



Drawing only specific Nodes/Edges

We can draw only specific nodes/edges by passing a list of nodes/edges to the drawing functions:

```
#draw only the nodes [3,4,5,6] and all edges
#connected to these nodes
G = nx.gnp_random_graph(10,0.3)
pos=nx.spring_layout(G)
nodes=[3,4,5,6]
edges = [e for e in G.edges() \
          if e[0] in nodes or e[1] in nodes]
nx.draw_networkx_nodes(G,pos,nodelist=nodes)
nx.draw_networkx_edges(G,pos,edgelist=edges)
```

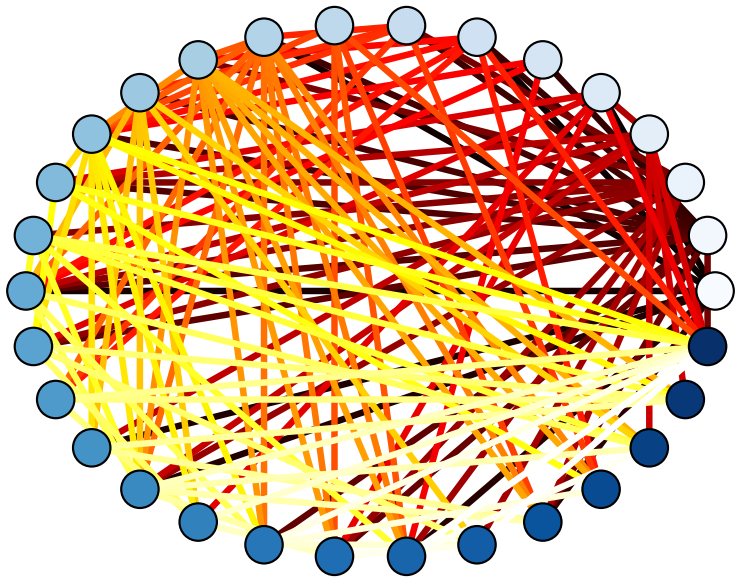
Colors (!)

NetworkX provides a nice layout as is but sometimes more colors just look better. The drawing functions expose a lot of matplotlib's options for color and transparency handling.

- Modify size and symbols of nodes via `node_size` and `node_shape`
- Modify node color via `node_color` and `cmap`
- Modify edge width and style via `width` and `style`
- Modify edge color via `edge_color` and `edge_cmap`
- Modify transparency via `alpha`

Example

```
G = nx.gnp_random_graph(30,0.3)
pos = nx.circular_layout(G)
blues = plt.cm.Blues
reds = plt.cm.hot
nx.draw_networkx_nodes(G,pos,\
    node_color=range(30),cmap=blues)
nx.draw_networkx_edges(G,pos,edge_color= \
    range(len(G.edges())),edge_cmap=reds,\
    width=3)
```

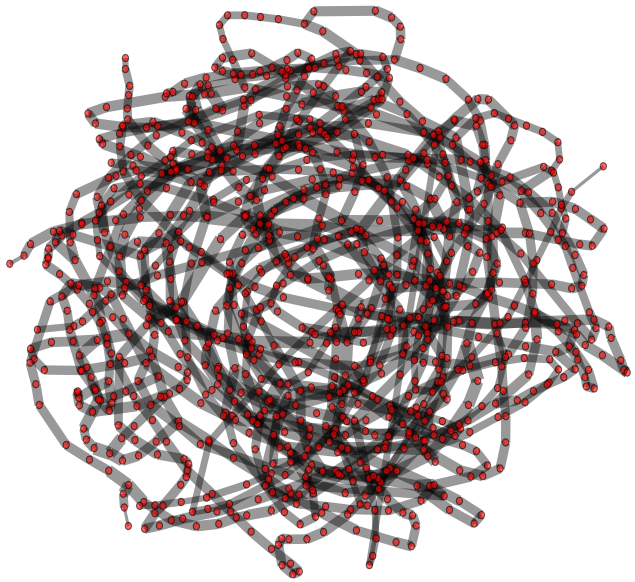


Plotting with Edge Attributes

```
G = nx.read_gpickle("medium_graph.gpickle")
pos = nx.spring_layout(G)
nx.draw_networkx_edges(G,pos,edgecolor= \
    'DarkSlateGray',alpha=0.4)
nx.draw_networkx_nodes(G,pos,color='FireBrick',\
    alpha=0.6,node_size=10,linewidths=0.5)
```

Plotting with Edge Attributes

```
G = nx.read_gpickle("medium_graph.gpickle")
pos = nx.spring_layout(G)
nx.draw_networkx_edges(G,pos,edgecolor= \
    'DarkSlateGray',alpha=0.4)
nx.draw_networkx_nodes(G,pos,color='FireBrick',\
    alpha=0.6,node_size=10,linewidths=0.5)
```



Plotting with Edge Attributes

```
G = nx.read_gpickle("medium_graph.gpickle")
pos = {}
for k in G.node.keys():
    pos[k] = (G.node[k]['x'], G.node[k]['y'])
nx.draw_networkx_edges(G,pos,edgecolor= \
    'DarkSlateGray',alpha=0.4)
nx.draw_networkx_nodes(G,pos,color='FireBrick',\
    alpha=0.6,node_size=10,linewidths=0.5)
```

Plotting with Edge Attributes

```
G = nx.read_gpickle("medium_graph.gpickle")
pos = {}
for k in G.node.keys():
    pos[k] = (G.node[k]['x'], G.node[k]['y'])
nx.draw_networkx_edges(G,pos,edgecolor= \
    'DarkSlateGray',alpha=0.4)
nx.draw_networkx_nodes(G,pos,color='FireBrick',\
    alpha=0.6,node_size=10,linewidths=0.5)
```