Mở đầu về thư viện NetworkX

Đỗ Xuân Anh
Trần Lưu Thái Phong
Hoàng Anh Quân
Đặng Đình Tài
Hoàng Thảo

Ngày 24 tháng 2 năm 2019

Đại học Khoa học Tự nhiên - ĐHQGHN

Table of contents

Giới thiệu về thư viện NetworkX

Thư viện NetworkX là gì?

Thư viện NetworkX là gì?

- Một thư viện Python dùng để xây dựng và thực hiện thao tác liên quan đến mạng lưới.

Thư viện NetworkX là gì?

- Một thư viện Python dùng để xây dựng và thực hiện thao tác liên quan đến mạng lưới.
- Một thư viện mã nguồn mở, được phát hành vào tháng 4 năm 2005.

Thư viện NetworkX là gì?

- Một thư viện Python dùng để xây dựng và thực hiện thao tác liên quan đến mạng lưới.
- Một thư viện mã nguồn mở, được phát hành vào tháng 4 năm 2005.
- https://networkx.github.io

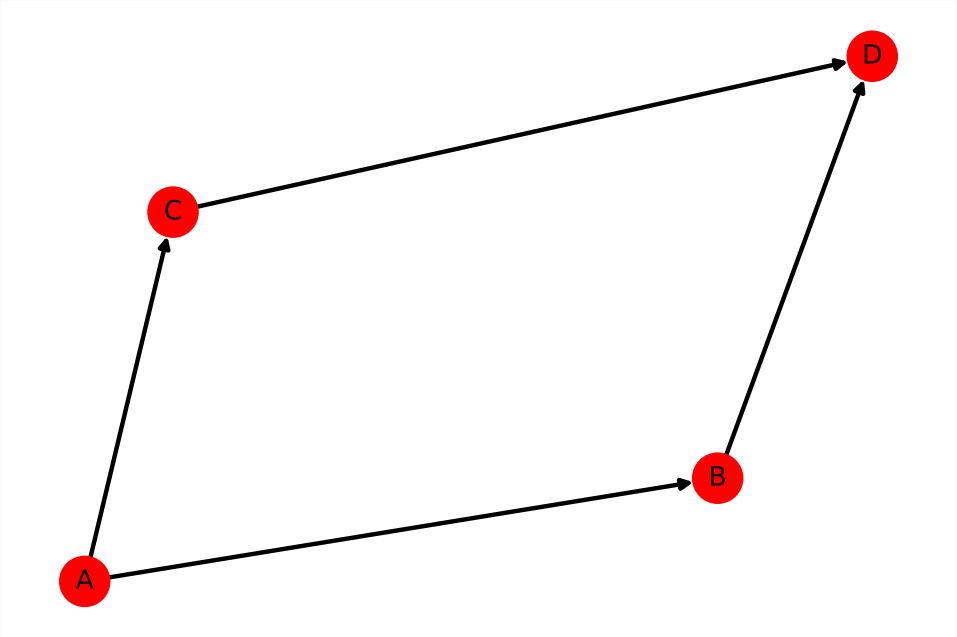
Các thao tác cơ bản với đồ thị

Tạo đồ thị, thêm đỉnh & cạnh

```
1  import networkx as nx
2  import matplotlib.pyplot as plt
3
4  #declare a directed graph
5  G = nx.DiGraph()
6
7  #add edges with different attributes
8  G.add_edge('A','B',capa=5)
9  G.add_edge('C','D',cost=10)
10
11 #add edges at a time
12 edge_list = [['A','B',2],['B','D',5],
13               ['A','C',4],['C','D',6]]
14 G.add_weighted_edges_from(edge_list,weight='distance')
15
16 #shortest path
17 path = nx.shortest_path(G,'A','D',weight='distance')
18 dis = nx.shortest_path_length(G,'A','D',weight='distance')
```

Vẽ đồ thị

```
#draw the graph
plt.figure()
options = {'node_color':'red','node_size':500, 'width':2}
nx.draw(G, with_labels=True, **options)
plt.savefig('graph.pdf')
```



Hình 1: graph.pdf

Node and Edge

```
#adding nodes
g.add_node(2)
g.add_node(4)

#adding an edge
g.add_edge(1,2)

#adding edges
g.add_edges_from([(1,2),(5,6),(3,4)])
g.add_nodes_from(range(1,10))
```

Write edge information in a file

```
g=nx.Graph()

#write a file txt
g.add_edge(1,2,weight=10)
g.add_edge(2,3,weight=20)
g.add_edge(3,4,weight=30)

nx.write_edgelist(g,'edgelist2.txt')
```

Create A Graph by reading Edge List from file

```
g=nx.read_edgelist('datanodes.txt',create_using=nx.Graph(),
    nodetype=int)
```

Removing nodes and edges

```
#removing a node
g.remove_node(1)

#removing nodes
g.remove_nodes_from(range(10))

#removing egdes
g.remove_edges_from([(1,2),(3,4)])
```

Making a complete graph

```
#making a completegraph
g=nx.complete_graph(100)

#print nodes
print (g.nodes())

#print egdes
print (g.edges())

#order
print (g.order())

#size
print (g.size())
```

Drawing a graph

```
g=nx.Graph()
g.add_edge(0,1)

#code of drawing
nx.draw_networkx(g)
```

Adding the position of nodes

```
G1=nx.Graph()

G1.add_node(0,pos=(1,1))
G1.add_node(1,pos=(1,2))
G1.add_node(2,pos=(2,4))
G1.add_node(4,pos=(3,4))
```

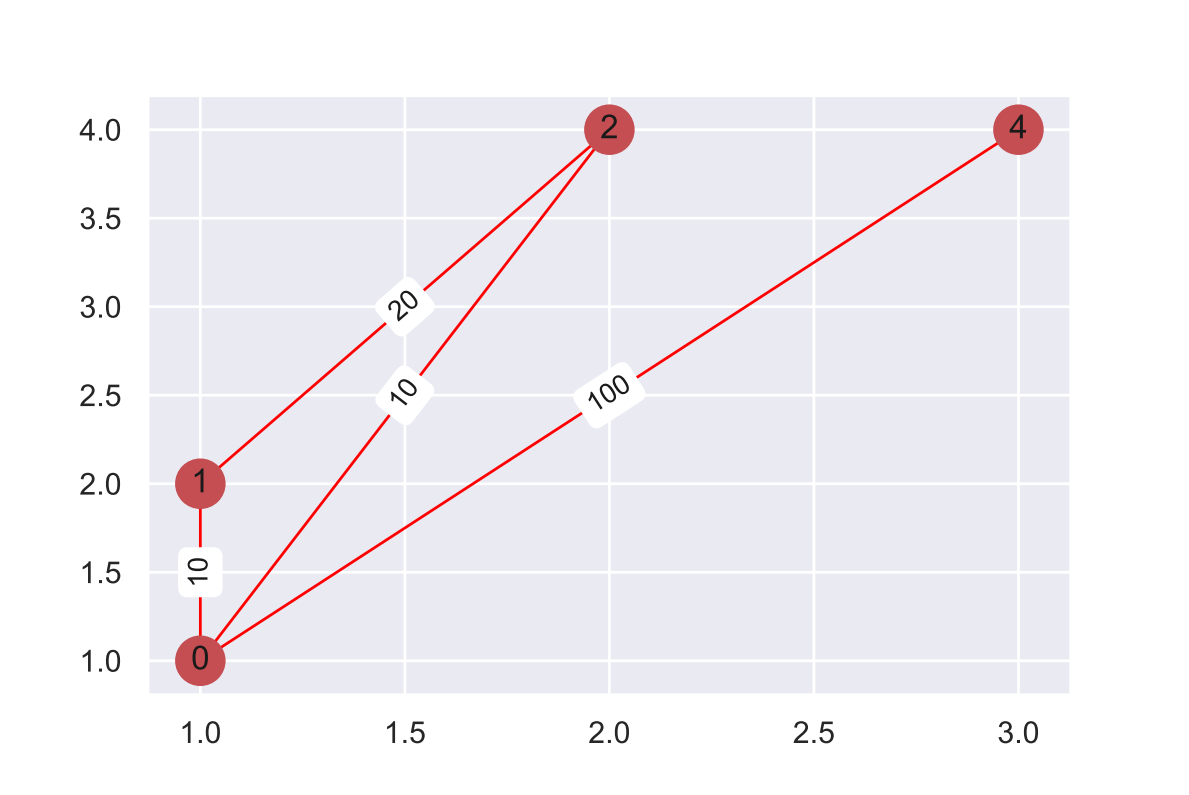
Adding the weight of edges

```
G1.add_edge(0,1,weight=10)
G1.add_edge(1,2,weight=20)
G1.add_edge(0,2,weight=10)
G1.add_edge(0,4,weight=100)
```

Drawing a graph

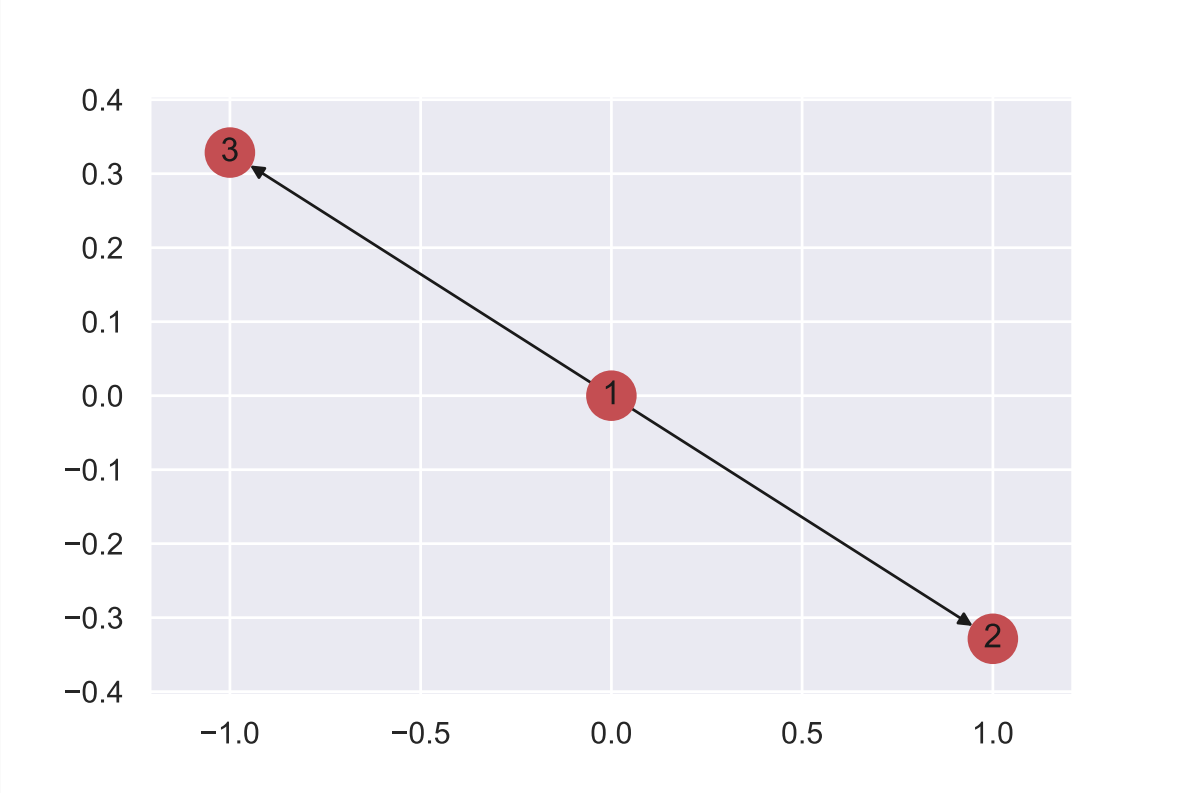
```
G1.add_node(0,pos=(1,1))
G1.add_node(1,pos=(1,2))
G1.add_node(2,pos=(2,4))
G1.add_node(4,pos=(3,4))

G1.add_edge(0,1,weight=10)
G1.add_edge(1,2,weight=20)
G1.add_edge(0,2,weight=10)
G1.add_edge(0,4,weight=100)

weight=nx.get_edge_attributes(G1,'weight')
pos=nx.get_node_attributes(G1,'pos')
plt.figure()

nx.draw_networkx(G1,pos,edge_color='red')
nx.draw_networkx_edge_labels(G1,pos,edge_labels=weight)
```

Drawing a graph



Digraph

```
1 G1=nx.DiGraph()
2 G1.add_edge(1,2)
3 G1.add_edge(1,3)
```



Bài toán đường đi ngắn nhất - Shortest path

Shortest path

Tìm các đường đi ngắn nhất giữa các nút trên đồ thị và độ dài của chúng

3.45 Shortest Paths

Compute the shortest paths and path lengths between nodes in the graph.

These algorithms work with undirected and directed graphs.

shortest_path(G[, source, target, weight, ...])	Compute shortest paths in the graph.
all_shortest_paths(G, source, target[, ...])	Compute all shortest paths in the graph.
shortest_path_length(G[, source, target, ...])	Compute shortest path lengths in the graph.
average_shortest_path_length(G[, weight, method])	Return the average shortest path length.
has_path(G, source, target)	Return *True* if *G* has a path from *source* to *target*.

3.45.1 networkx.algorithms.shortest_paths.generic.shortest_path

shortest_path (*G*, *source=None*, *target=None*, *weight=None*, *method='dijkstra'*)

Compute shortest paths in the graph.

Bảng 1: NetworkX reference

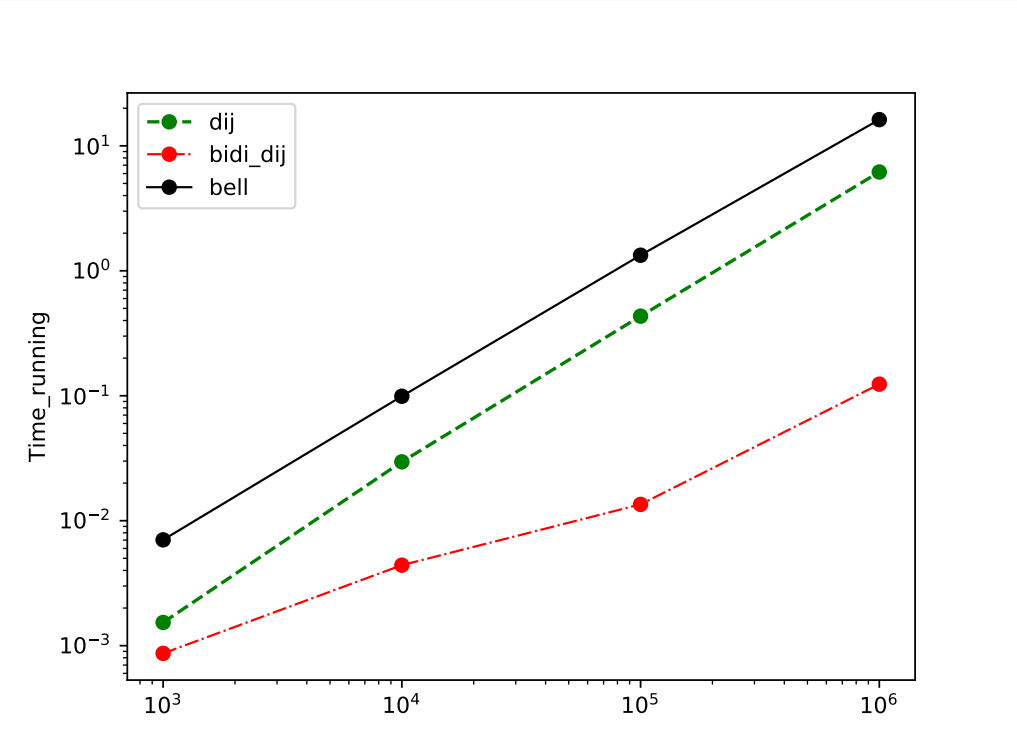
Shortest path

- nx.shortest_path(G,source,target,weight,method='dijkstra')
- nx.shortest_path(G,source,target,weight,method='bellman-ford')
- nx.dijkstra_path(G,source,target,weight)
- nx.bidirectional_dijkstra(G,source,target,weight)
- nx.bellman_ford_path(G,source,target,weight)

Shortest path

So sánh thời gian chạy của 3 thuật toán với $G = (n, 2n)$

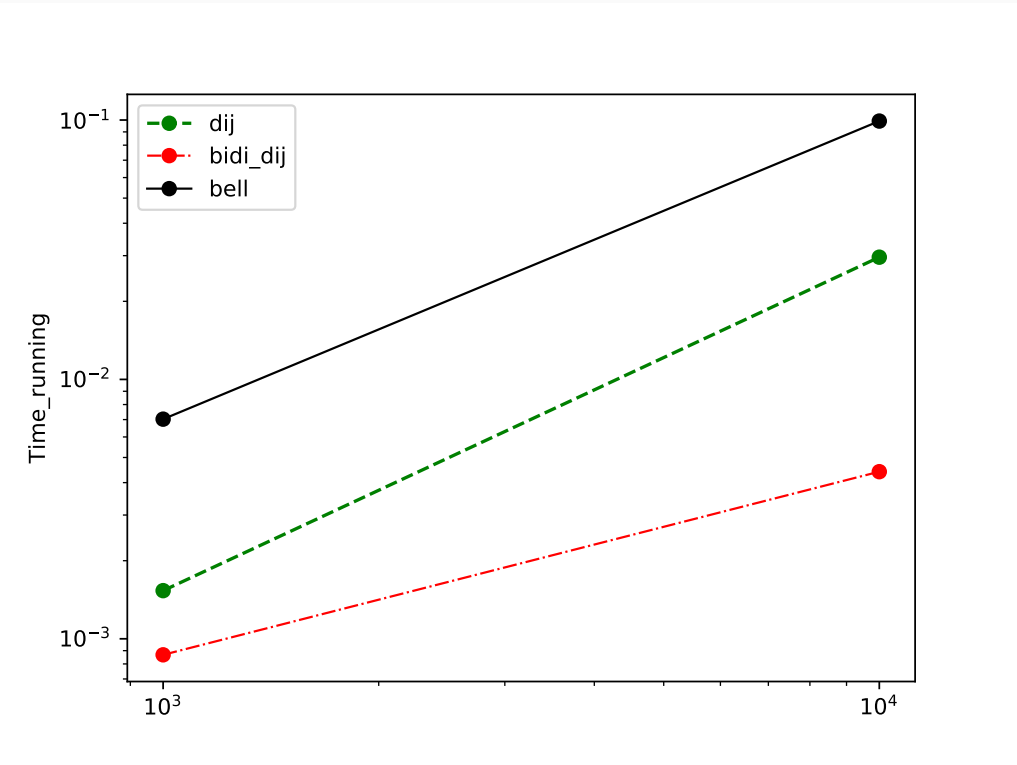
	dijkstra	bellman ford	bidirectional dijstra
10^3	0.0015	0.0070	0.00086
10^4	0.0295	0.0991	0.0044
10^5	0.4335	1.332	0.0135
10^6	6.1739	16.204	0.1237



Shortest path

So sánh thời gian chạy của 3 thuật toán với $G = (n, n\sqrt{n})$

	dijkstra	bellman ford	bidirectional dijstra
10^3	0.0195	0.1172	0.0114
10^4	0.472	5.622	0.144



Bài toán luồng cực đại - Maximum flow

Thuật toán

```
flow_value, flow_dict = nx.maximum_flow(G,
                                        start_node,end_node,
                                        flow_func=None)
```

So sánh thời gian thực thi thuật toán

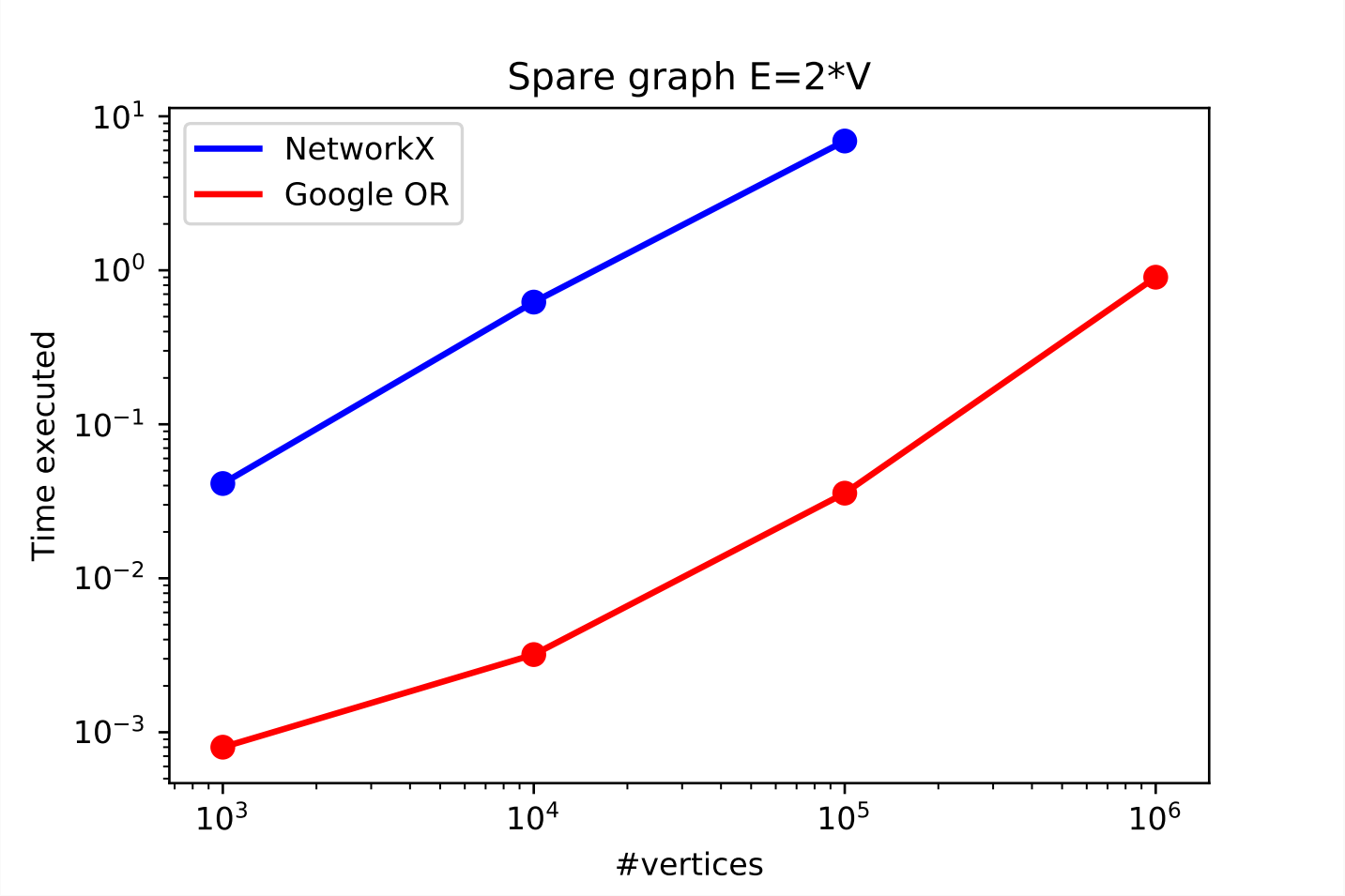
STT	Số đỉnh	NetworkX	Google OR tools	Nhanh hơn
1	1000	0.04124	0.0008	51 lần
2	10000	0.6216	0.003195	194 lần
3	100000	6.911	0.03570	193 lần
4	1000000	None	0.9019	∞

Bảng 2: Đồ thị với $E = 2V$

STT	Số đỉnh	NetworkX	Google OR tools	Nhanh hơn
1	1000	0.2836	0.002904	97 lần
2	10000	11.77	0.1636	71 lần

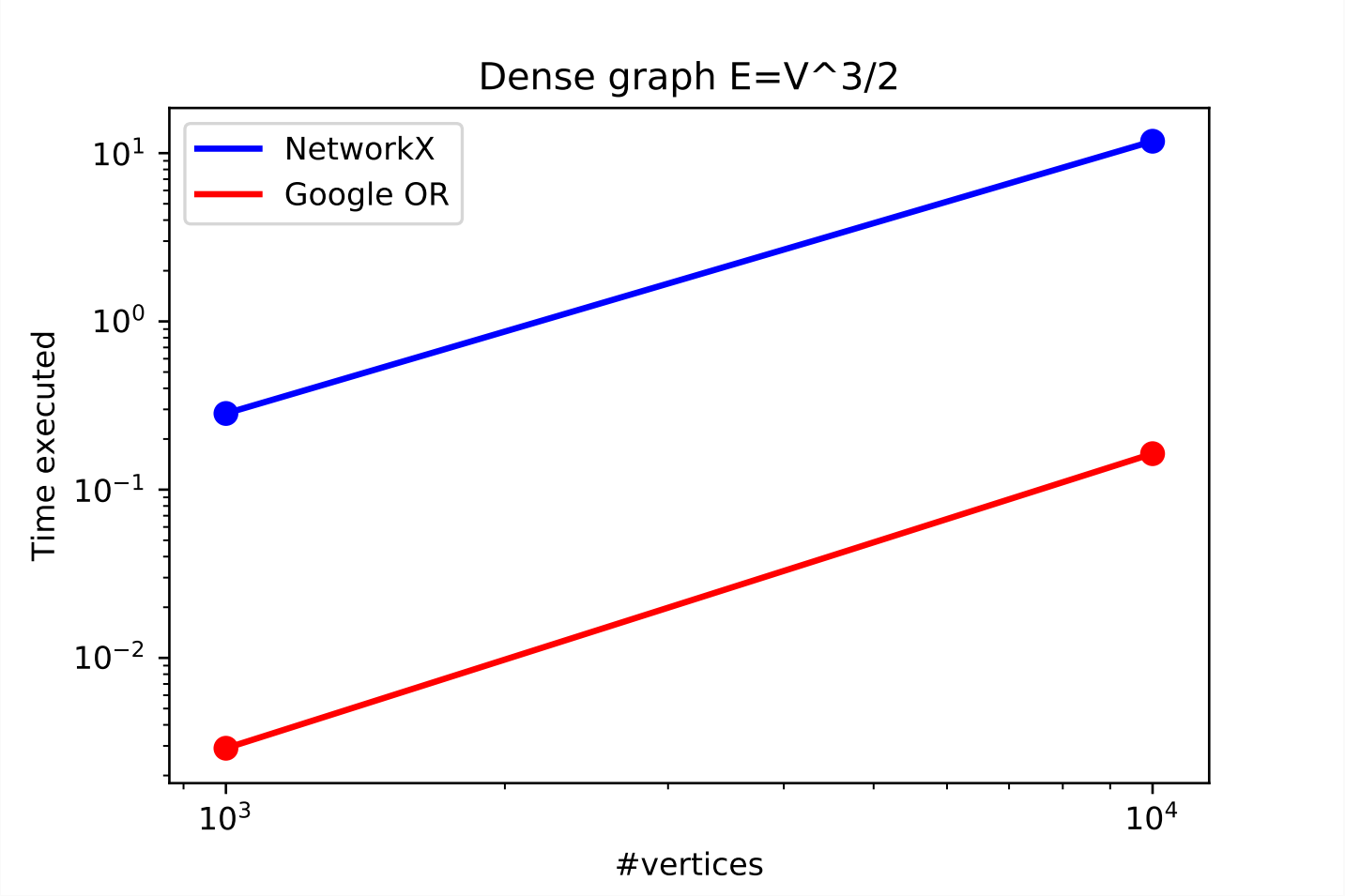
Bảng 3: Đồ thị với $E = V\sqrt{V}$

So sánh thời gian thực thi thuật toán



Hình 2: Đồ thị với $E = 2V$

So sánh thời gian thực thi thuật toán



Hình 3: Đồ thị với $E = V\sqrt{V}$

Các tùy chọn khác cho flow_func

```
from networkx.algorithms.flow import (
        shortest_augmenting_path,
        preflow_push,
        dinitz,
        boykov_kolmogorov)
```

```
flow_value, flow_dict = nx.maximum_flow(G,
                                start_node,end_node,
                                flow_func=None)
```

- shortest_augmenting_path
- preflow_push
- dinitz
- boykov_kolmogorov