Programação II
+
Estruturas de Dados para Bioinformática

Visualização de grafos

Hugo Pacheco

DCC/FCUP
22/23

Grafos

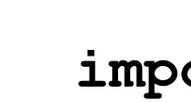
- Umas aulas atrás (*NetworkX*):
    - Representar redes de dados interligados como grafos
        - Utilização de algoritmos de grafos
- Esta aula (*NetworkX* + *Graphviz*):
    - Visualização de grafos
    - Mais exemplos

Grafos

- Visualização de grafos é um problema complexo
  - Minimizar sobreposições, maximizar distâncias, etc
- Vamos explorar visualização de grafos em Python
  - Funcionalidades base do *NetworkX*
  - Integração com *Graphviz*
- Não vamos ver ferramentas mais completas de visualização de grafos
  - E.g., *Citoscape*, *Gephi*

NetworkX (*draw*)

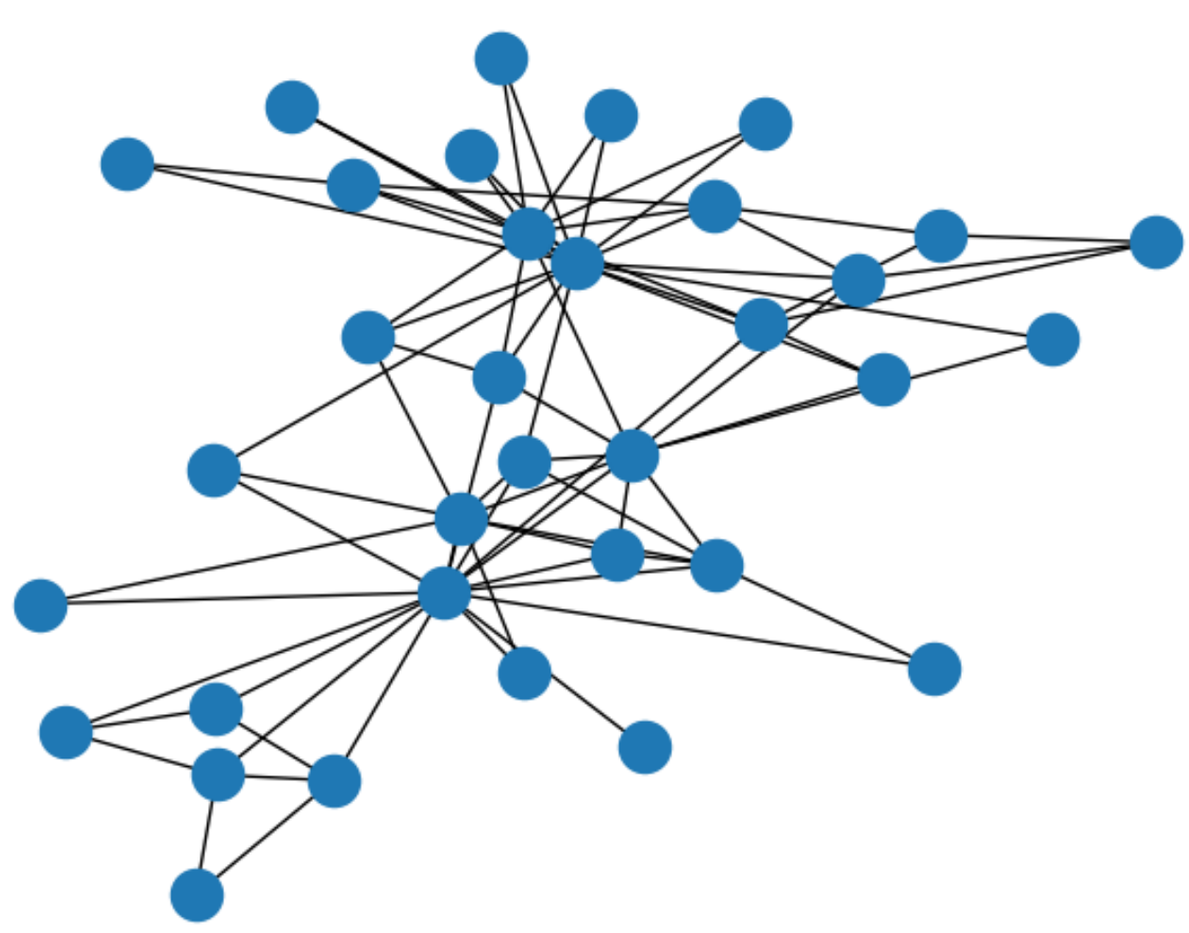
- Visualizar um grafo de exemplo



Se redesenhar o grafo, as posição mudam!

```
import networkx as nx
import matplotlib.pyplot as plt

g = nx.karate_club_graph()
print(g.nodes(data=True))
# [(0, {'club': 'Mr. Hi'}),...]
print(g.edges(data=True))
# [(0, 1, {'weight': 4}),...]
nx.draw(g)
plt.show()
```

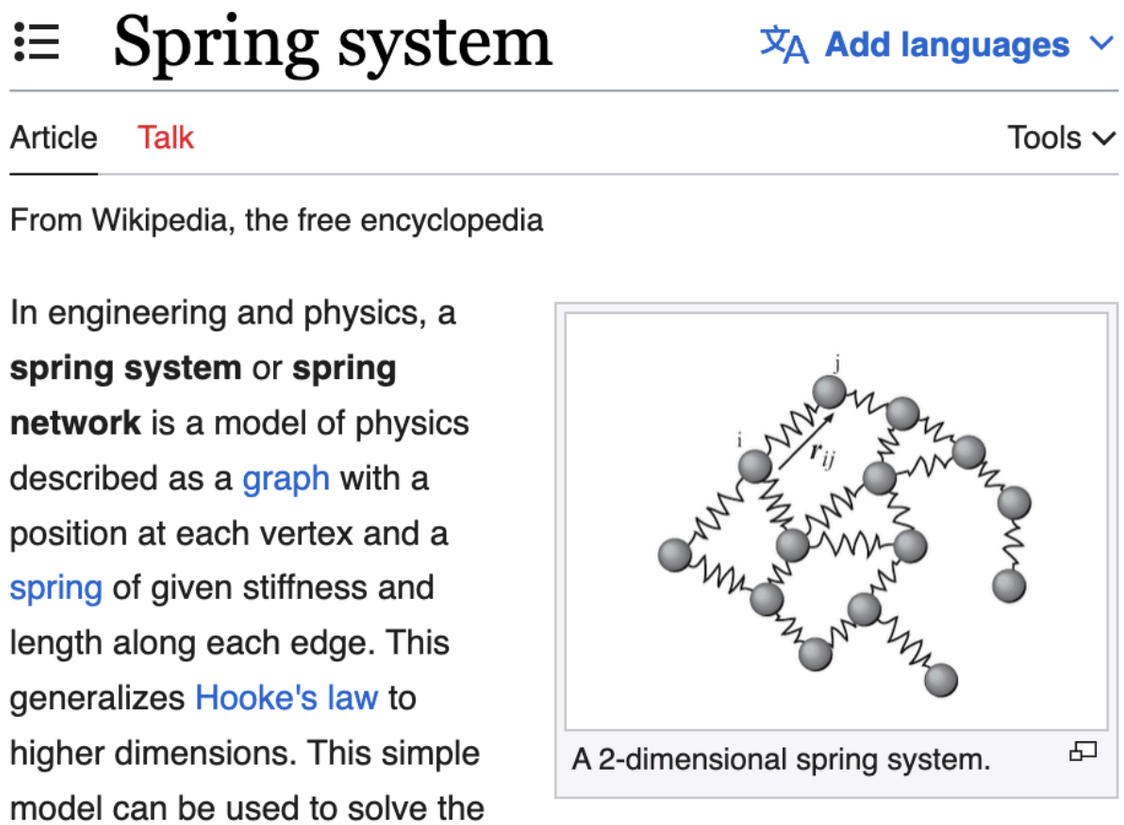


NetworkX (layout)

- Como é que o NetworkX decide como desenhar o grafo?
  - Utiliza um layout, e.g., um dicionário *{ nodo : Posição2D }*
- O default é o spring layout, que tende a desenhar nodos mais conectados mais próximos uns dos outros

```
l = nx.spring_layout(g)
print(l)
# {0: array([-0.24019787,
-0.34623371]), ...}
l = nx.spring_layout(g)
print(l)
# {0: array([-0.07914912,
0.46013368]), ...}

nx.draw(g,pos=l)
```



Spring system

Add languages

Article Talk Tools

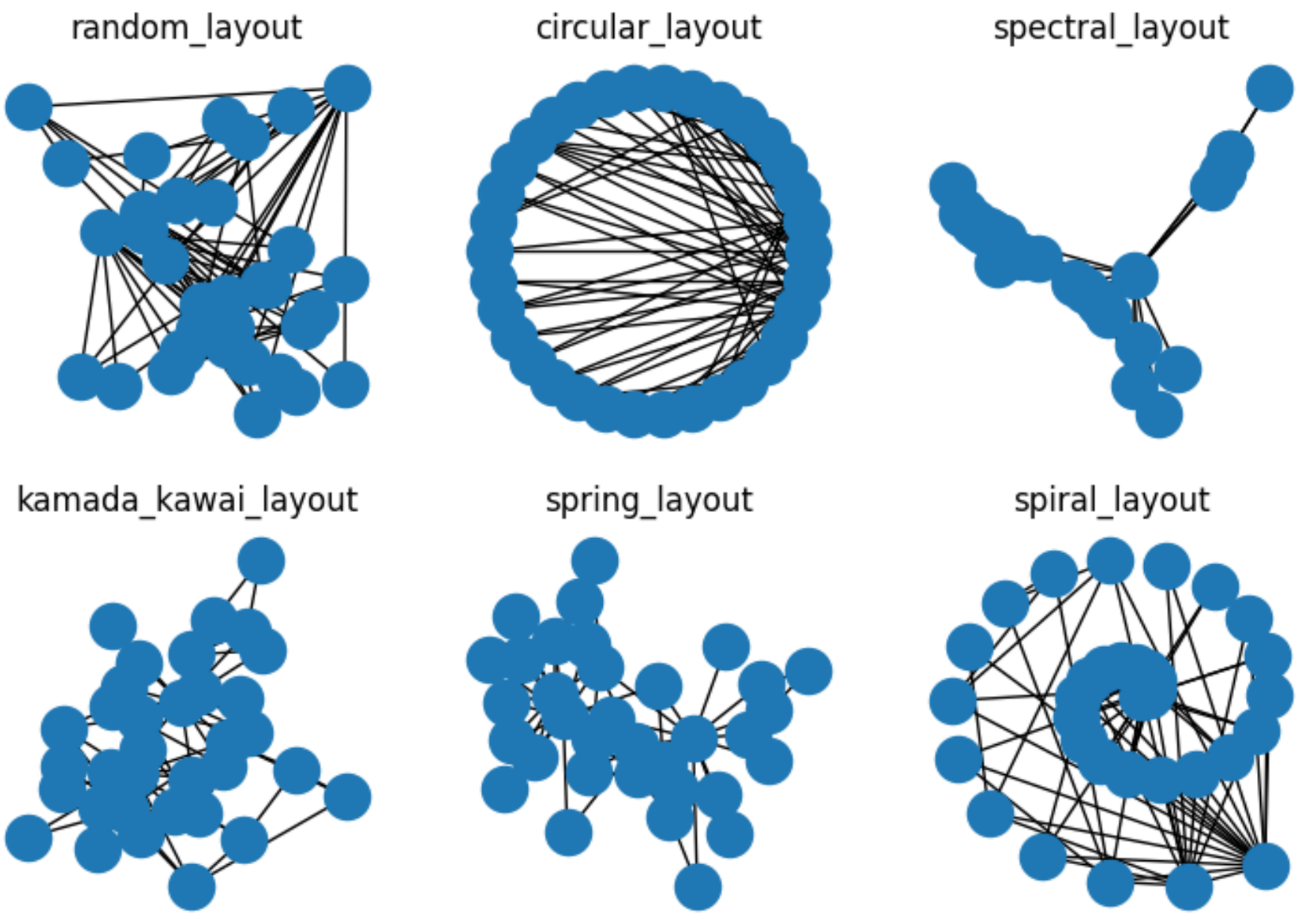
From Wikipedia, the free encyclopedia

In engineering and physics, a **spring system** or **spring network** is a model of physics described as a graph with a position at each vertex and a spring of given stiffness and length along each edge. This generalizes Hooke's law to higher dimensions. This simple model can be used to solve the

A 2-dimensional spring system.

NetworkX (layout)

- Há vários layouts disponíveis (documentação)



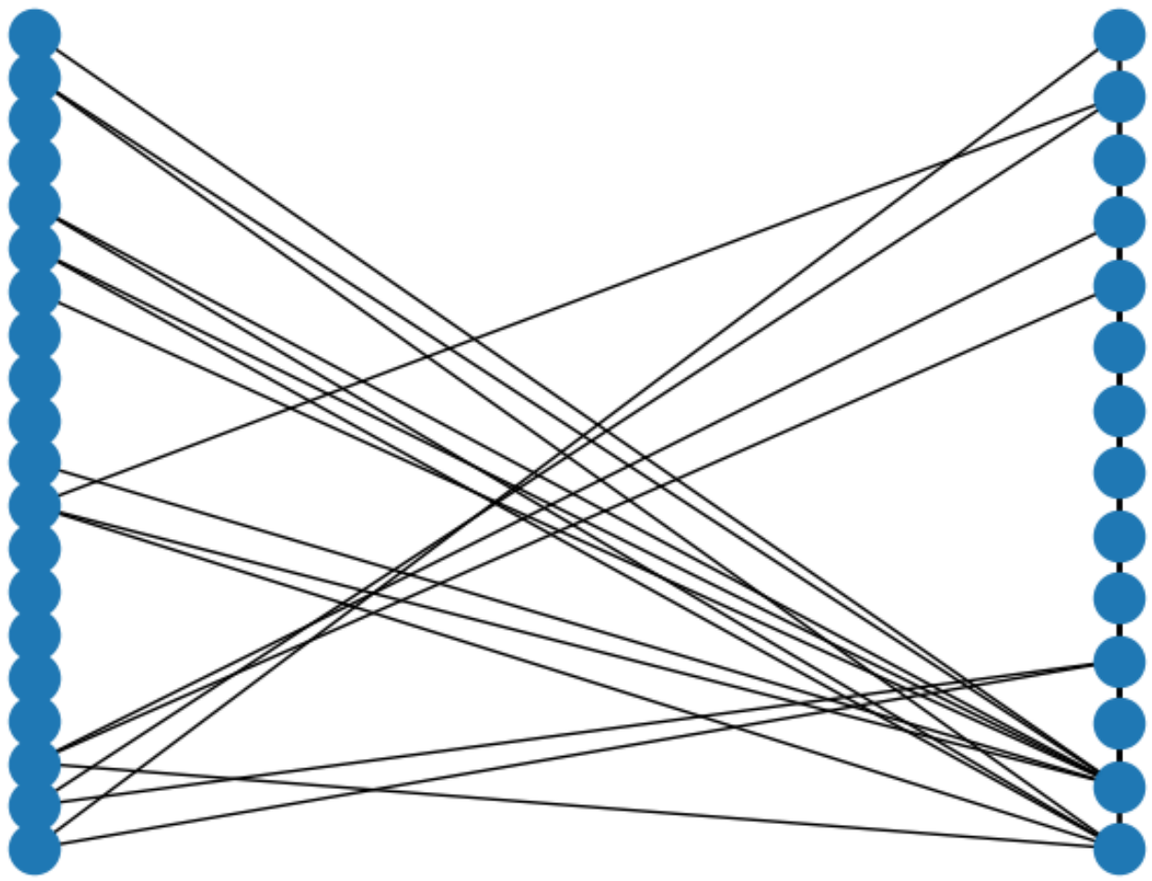
NetworkX (layout)

- O *bipartite_layout* desenha o grafo de acordo com duas partições

  - Temos que passar os nodos que constituem uma delas

```
g = nx.karate_club_graph()

ns = [n for n in g.nodes()
if n < 20 ]
nx.draw(g,pos=nx.bipartite
_layout(g,ns))
```

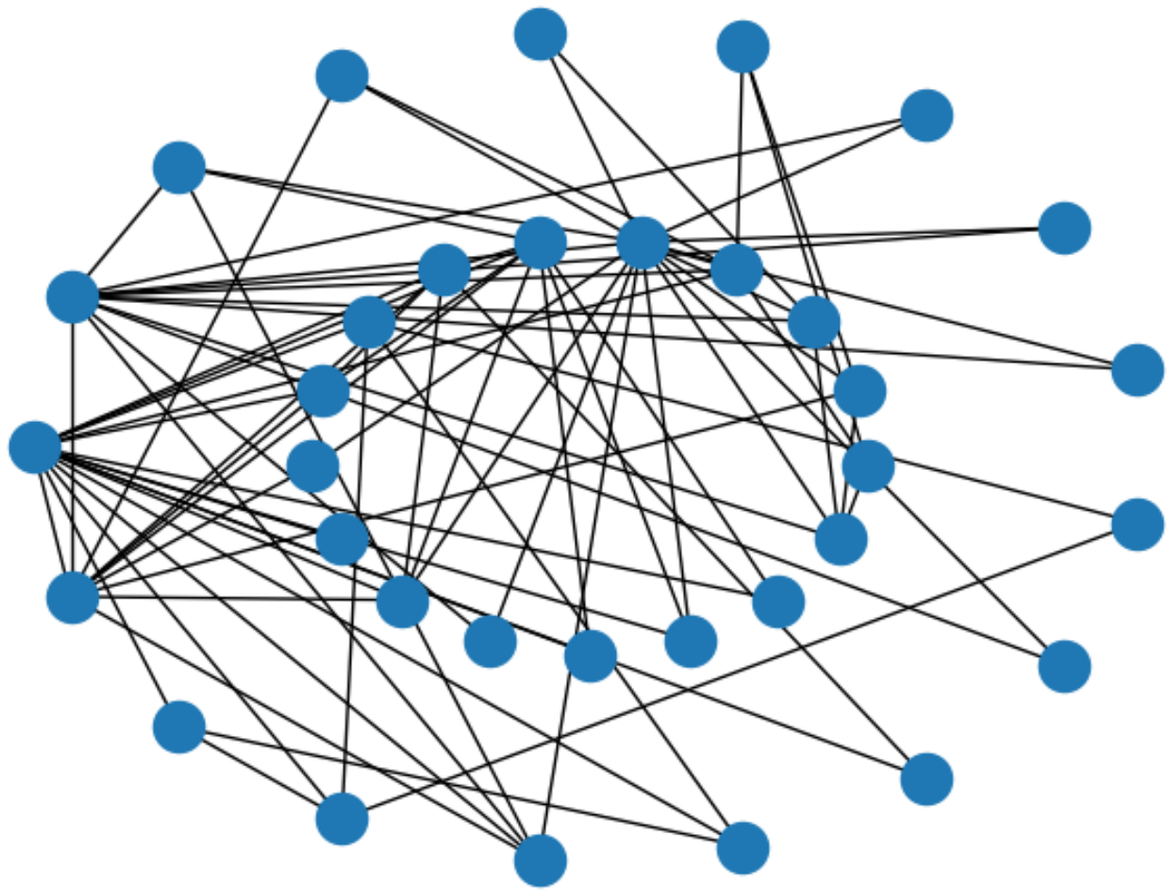


NetworkX (layout)

- O *shell_layout* desenha o grafo por níveis
  - Temos que passar uma lista de níveis, em que um nível é uma lista de nodos

```
g = nx.karate_club_graph()

ns = [[n for n in
g.nodes() if n % i == 0]
for i in range(1,3)]
nx.draw(g,pos=nx.shell_lay
out(g,ns))
```

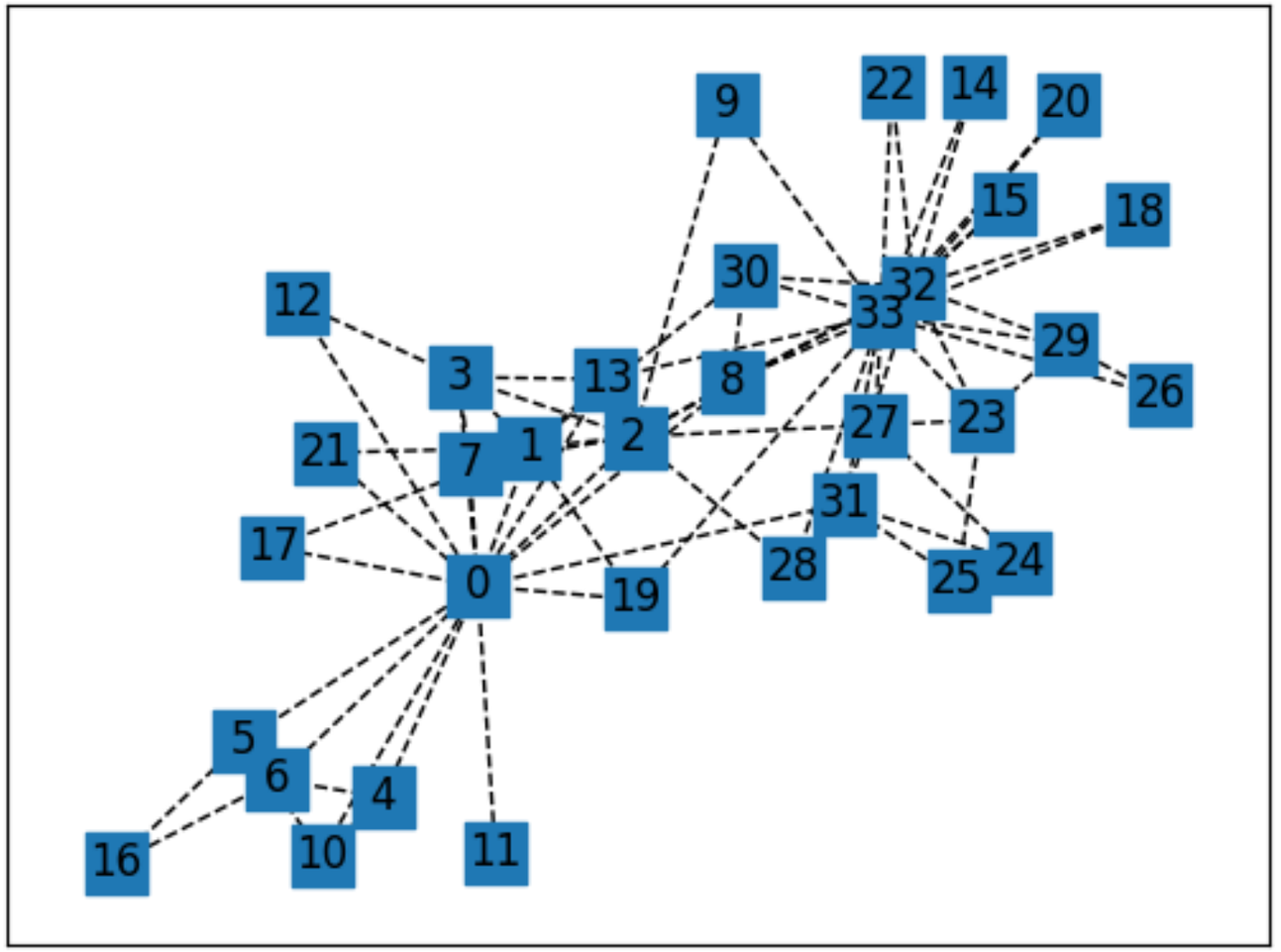


NetworkX (*draw_networkx*)

- Podemos costumizar melhor a visualização
  - e.g., formato de nodos, formato de arestas

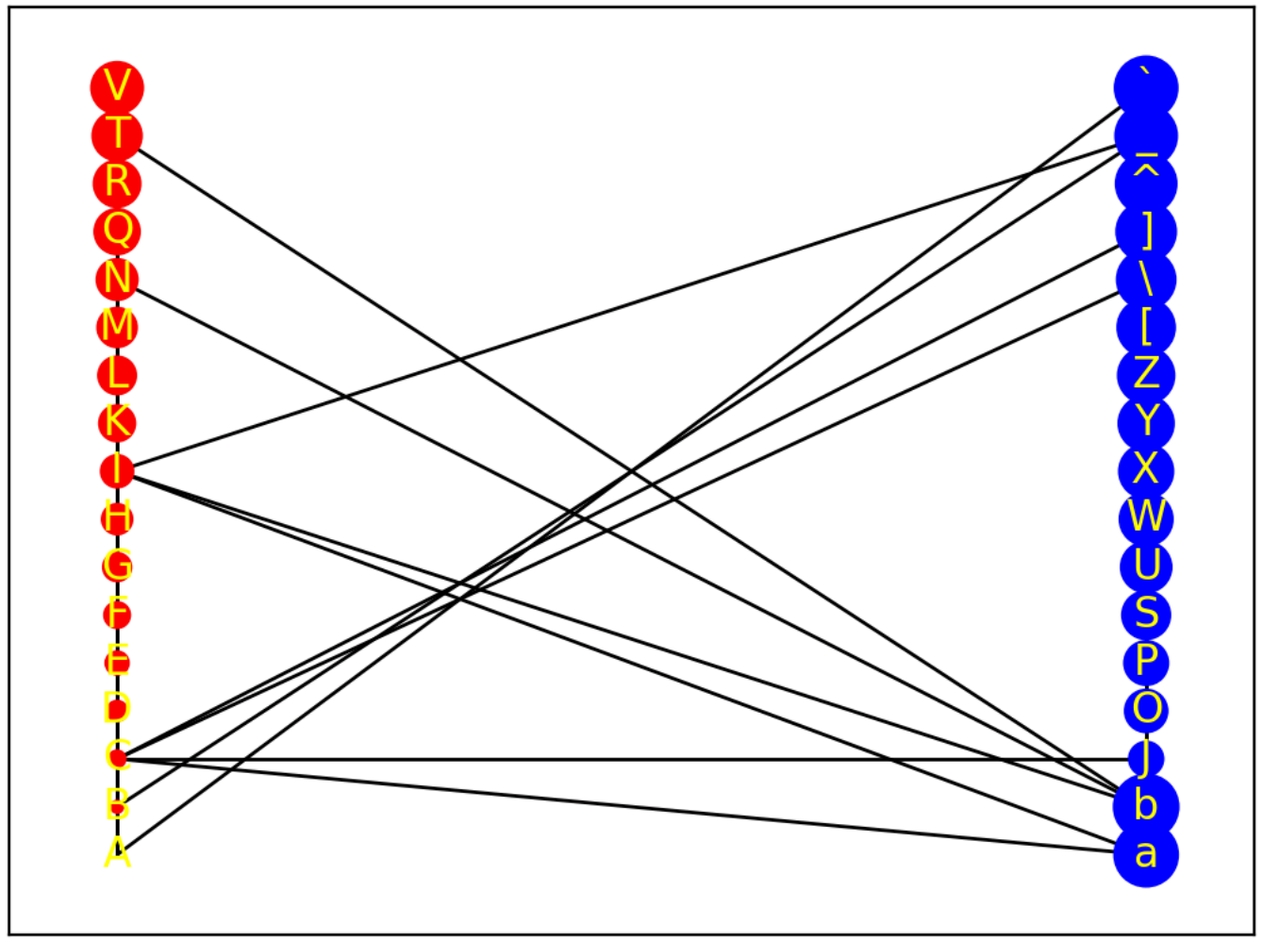
```
g = nx.karate_club_graph()

nx.draw_networkx(
 g
 ,node_shape='s'
 ,style="--")
plt.show()
```



NetworkX (*draw_networkx*)

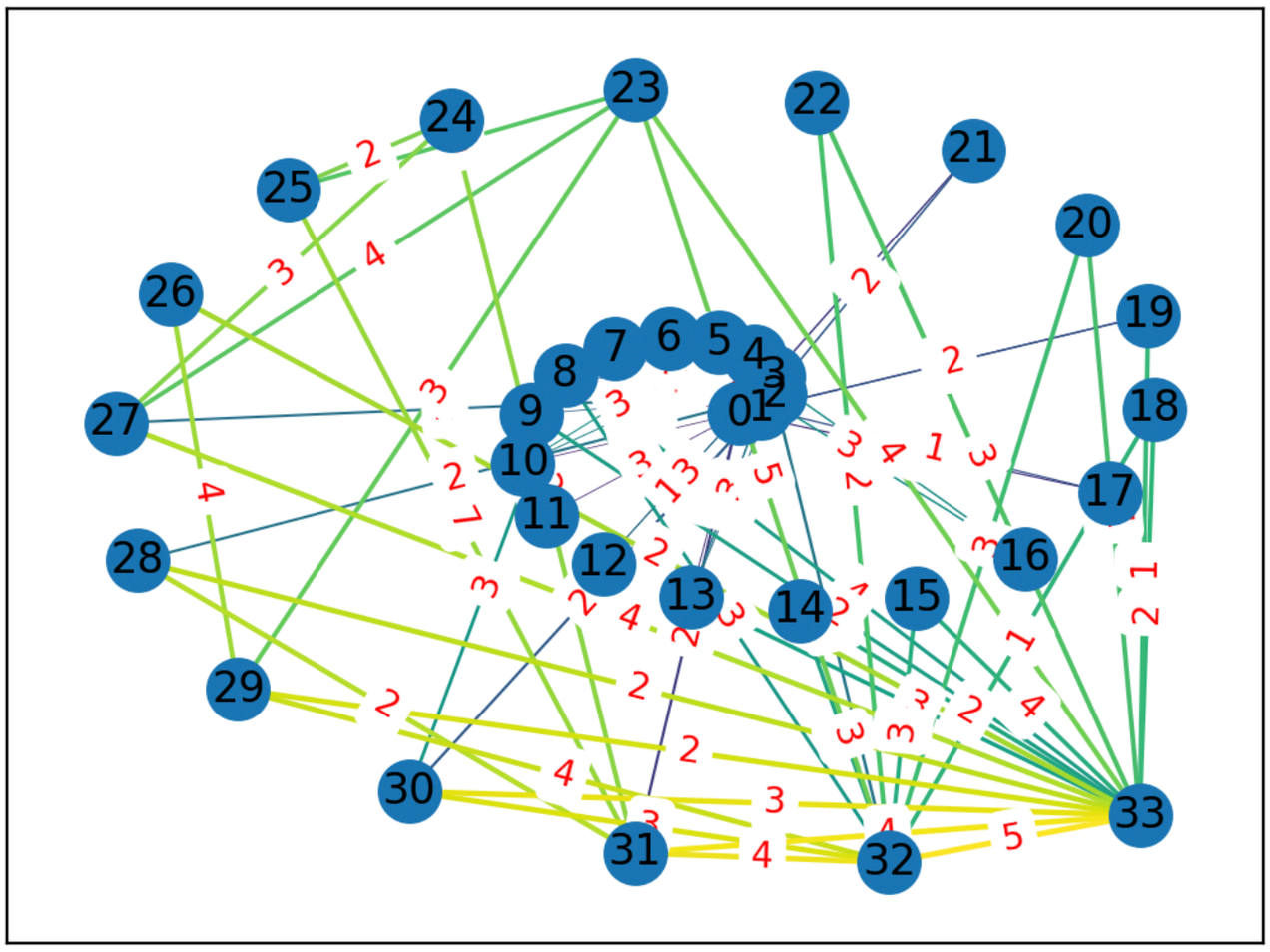
- Podemos costumizar melhor a visualização (por nodo)
  - e.g., labels, tamanho, cor




```
g = nx.karate_club_graph()
ns = g.nodes(data=True)
color = {'Mr. Hi':'red'
        ,'Officer':'blue'}
his = [n for n,i in ns
         if i['club']=='Mr. Hi']
colors = [ color[i['club']]
           for n,i in ns ]
ls = { n : chr(n+65) for n,i in ns }
szs = [10 * n for n in g.nodes()]
nx.draw_networkx(g,pos=nx.bipartite_layout(g,his),labels=ls,node_siz
e=szs,node_color=colors,font_color='yellow')
```

NetworkX (*draw_networkx*)

- Podemos costumizar melhor a visualização (por aresta)
  - e.g., labels, grossura, cor

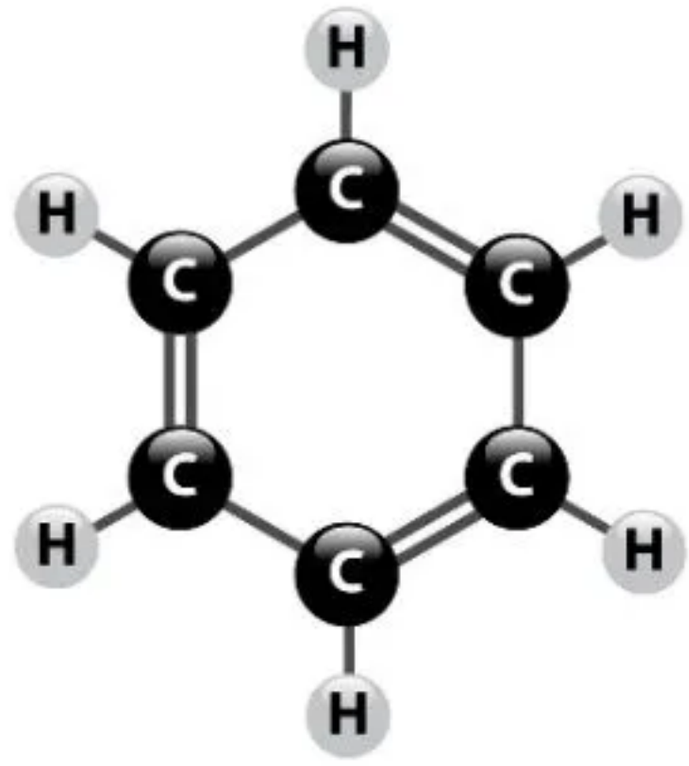



```
g = nx.karate_club_graph()
es = g.edges(data=True)
ws = [ (i+j)/40
       for i,j,d in es ]
cs = [ i for i
       in range(len(es)) ]
ls = { (i,j) : d['weight']
       for i,j,d in es }
ps = nx.spiral_layout(g)
nx.draw_networkx(g,pos=ps,width=ws,edge_color=cs)
nx.draw_networkx_edge_labels(g,pos=ps,edge_labels=ls,font_color='red')
```

Exemplo (Molécula)

- Desenhar uma molécula no formato MOL2 (mais detalhes), e.g., benzeno
  - nodos = átomos; arestas = ligações

```
5   @<TRIPOS>MOLECULE
6   benzene
7    12 12 0 0 0
8   SMALL
9   NO_CHARGES
10  ****
11  Any comment about the molecule goes here. No need for a pound sign he
12
13  @<TRIPOS>ATOM
14        1 C          -0.7600    1.1691   -0.0005 C.ar    1  BENZENE
15        2 C           0.6329    1.2447   -0.0012 C.ar    1  BENZENE
16        3 C           1.3947    0.0765    0.0004 C.ar    1  BENZENE
17        4 C           0.7641   -1.1677    0.0027 C.ar    1  BENZENE
18        5 C          -0.6288   -1.2432    0.0001 C.ar    1  BENZENE
19        6 C          -1.3907   -0.0751   -0.0015 C.ar    1  BENZENE
20        7 H          -1.3536    2.0792    0.0005 H       1  BENZENE
21        8 H           1.1243    2.2140   -0.0028 H       1  BENZENE
22        9 H           2.4799    0.1355   -0.0000 H       1  BENZENE
23       10 H           1.3576   -2.0778    0.0063 H       1  BENZENE
24       11 H          -1.1202   -2.2126   -0.0005 H       1  BENZENE
25       12 H          -2.4759   -0.1340   -0.0035 H       1  BENZENE
26  @<TRIPOS>BOND
27       1     1     2   ar
28       2     2     3   ar
```



Benzeno

C_6H_6

Exemplo (Molécula)

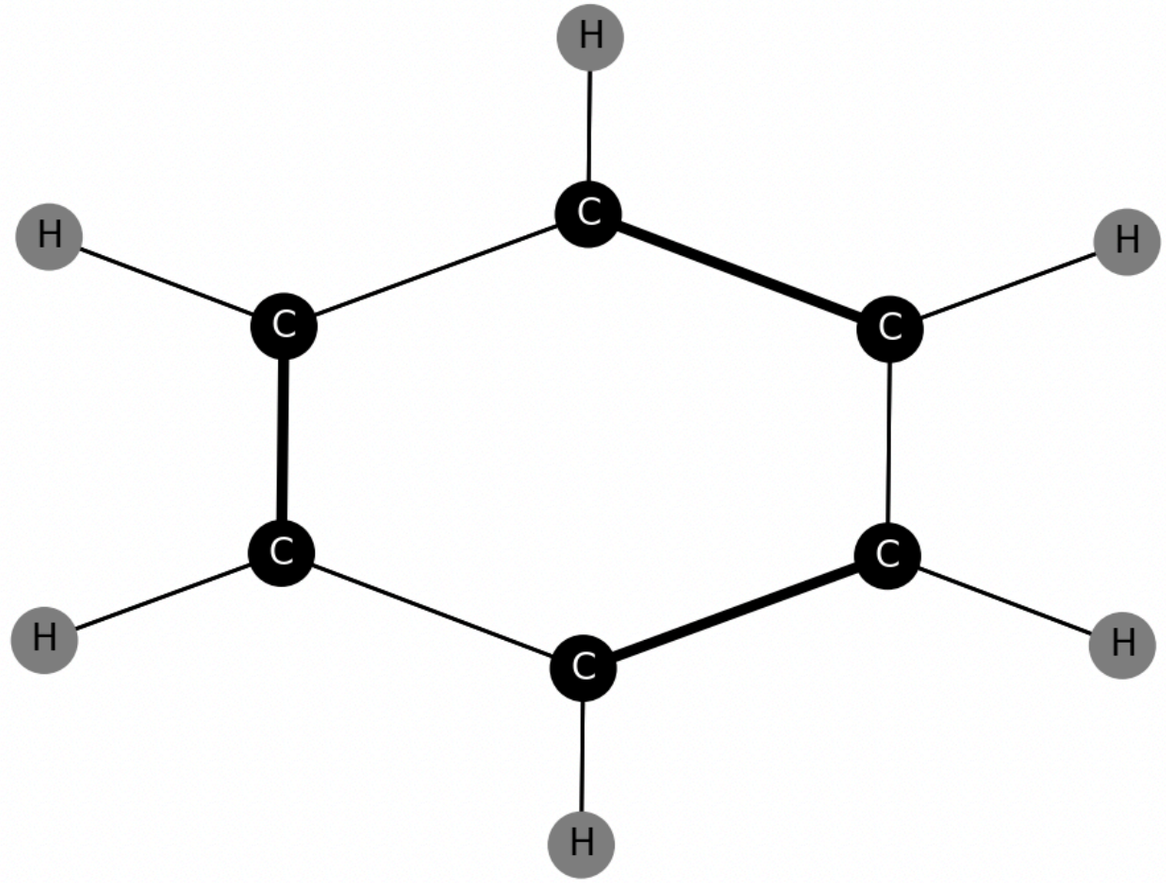
1. Ler ficheiro MOL2 retirado daqui

2. Converter <TRIPOS>ATOM num DataFrame

3. Converter <TRIPOS>BOND num DataFrame

4. Juntar ambos num grafo NetworkX

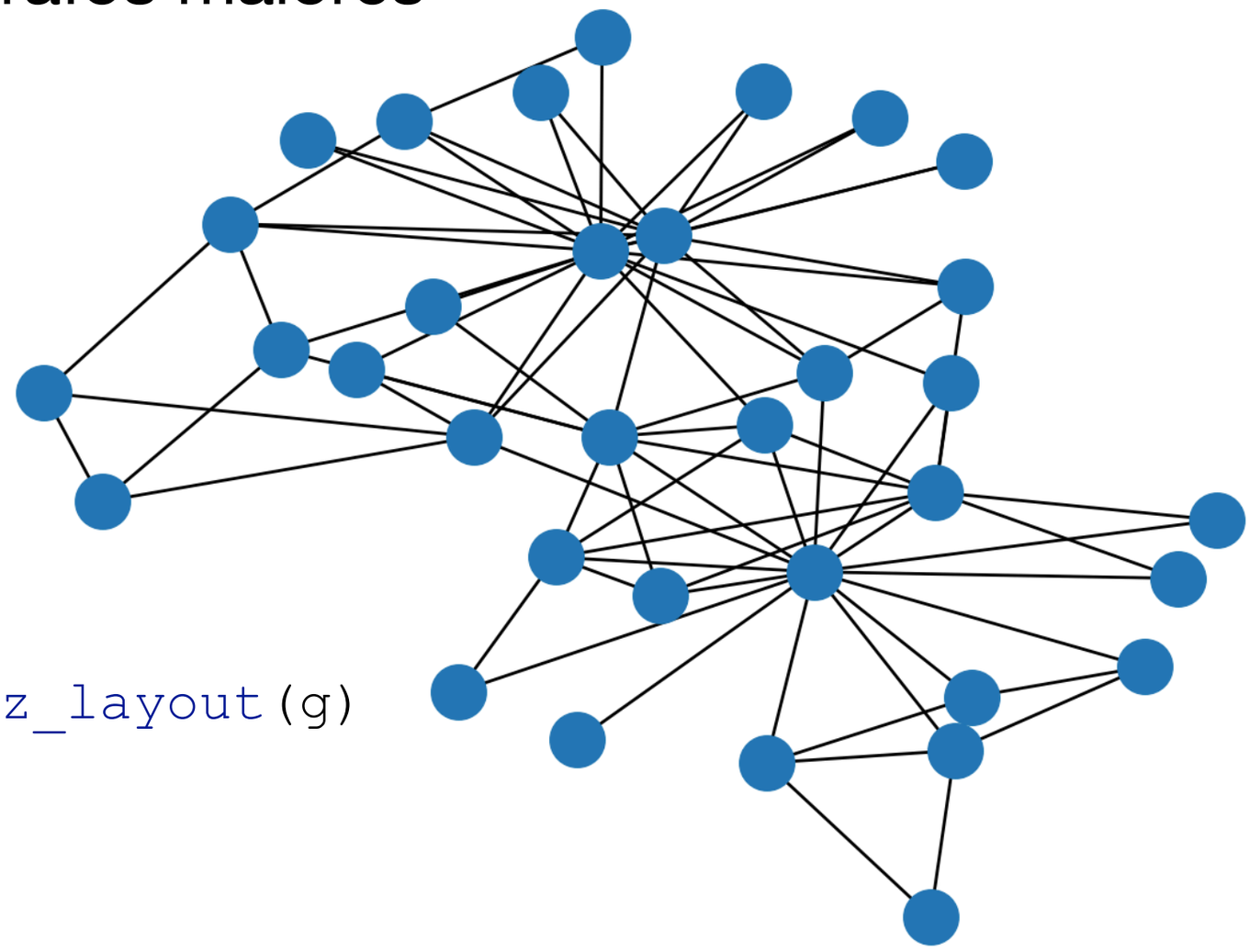
5. Desenhar nodos e arestas



NetworkX ← Graphviz

- Podemos adicionalmente utilizar layouts do Graphviz (via a biblioteca *pygraphviz*)

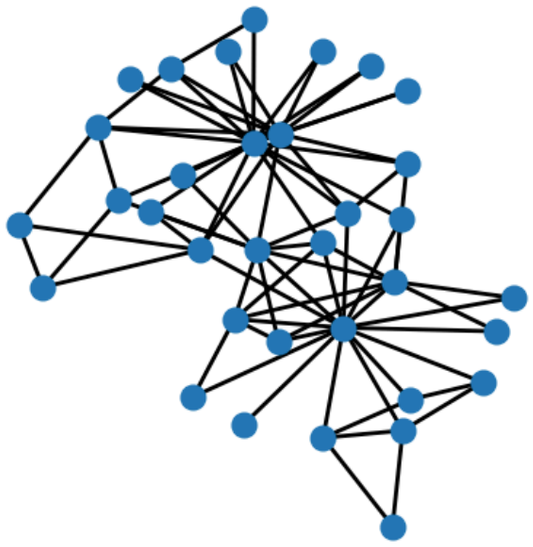
- Particularmente útil para grafos maiores



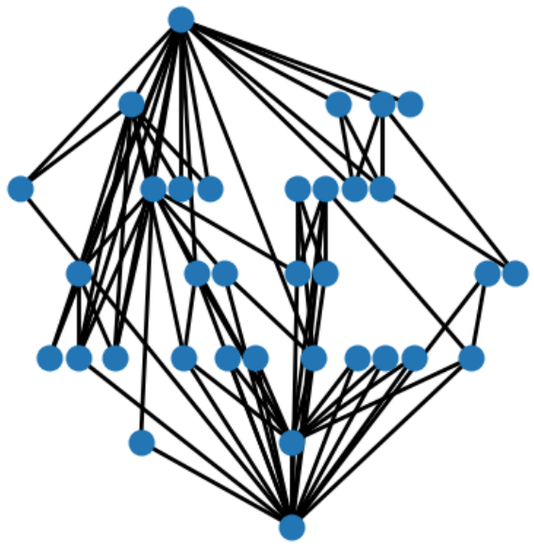
```
g = nx.karate_club_graph()
l = nx.nx_agraph.pygraphviz_layout(g)
nx.draw(g,pos=l)
```

NetworkX ← Graphviz

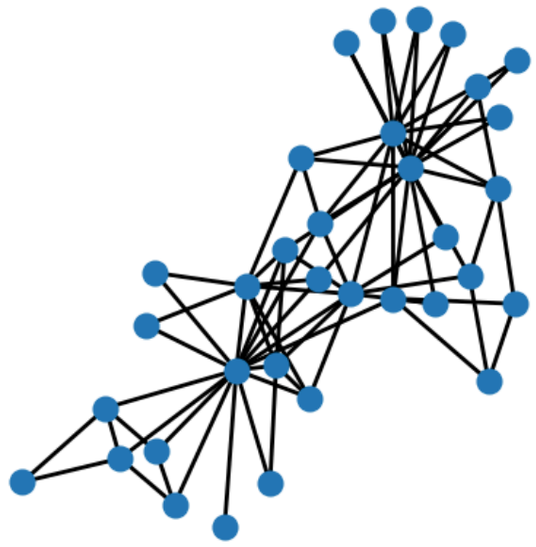
neato (default)



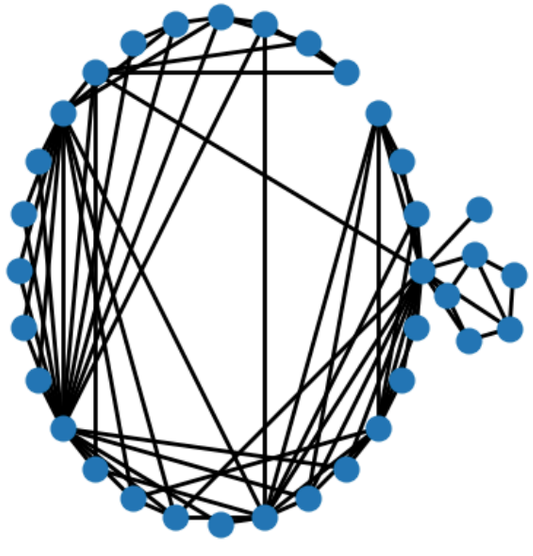
dot



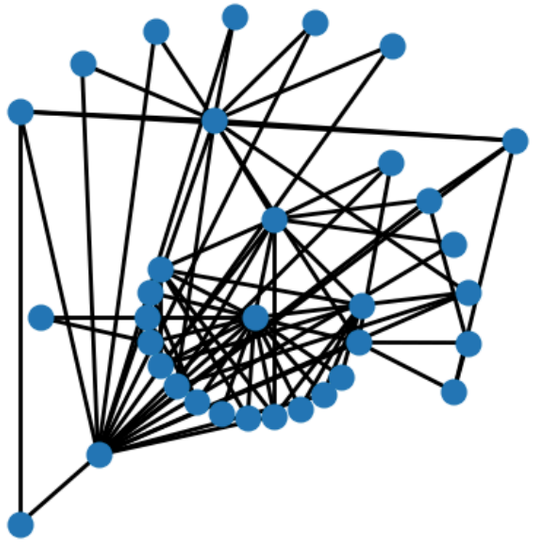
fdp



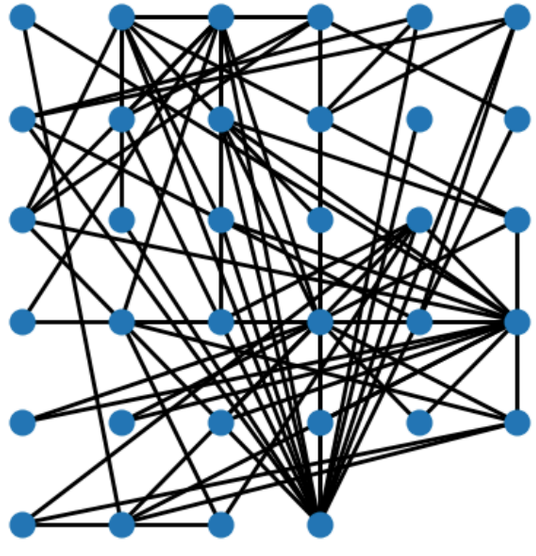
circo



twopi



osage



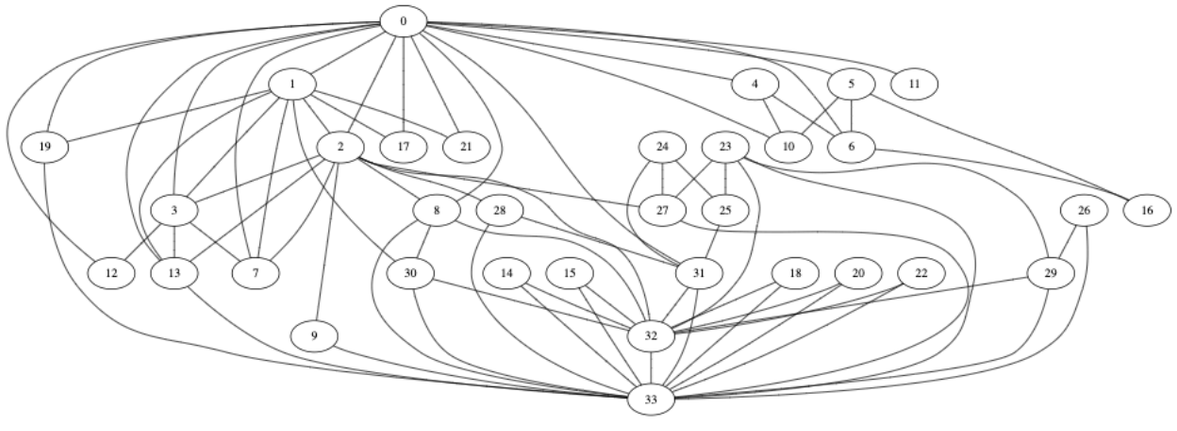
NetworkX → Graphviz

- Podemos converter um grafo no formato Graphviz, e.g., exportar para ficheiro DOT
- Particularmente útil para processamento adicional, e.g., https://edotor.net/

```
g = nx.karate_club_graph()
ag = nx.nx_agraph.to_agraph(ag)
ag.write('karate.dot')
```

Edotor Load Sample Download Engine: dot Graphviz Documentation Issues Copy Share Link

```
strict graph "Zachary's Karate Club" {
    graph [name="Zachary's Karate Club"];
    0   [club="Mr. Hi"];
    1   [club="Mr. Hi"];
    0 -- 1  [weight=4];
    2   [club="Mr. Hi"];
    0 -- 2  [weight=5];
    3   [club="Mr. Hi"];
    0 -- 3  [weight=3];
    4   [club="Mr. Hi"];
    0 -- 4  [weight=3];
    5   [club="Mr. Hi"];
    0 -- 5  [weight=3];
    6   [club="Mr. Hi"];
    0 -- 6  [weight=3];
    7   [club="Mr. Hi"];
    0 -- 7  [weight=2];
```



NetworkX → Graphviz

- O Graphviz é bastante mais configurável (documentação)
- Pygraphviz = interface programática
- Mais features:
  - clusters
  - formatação
  - layout
  - etc