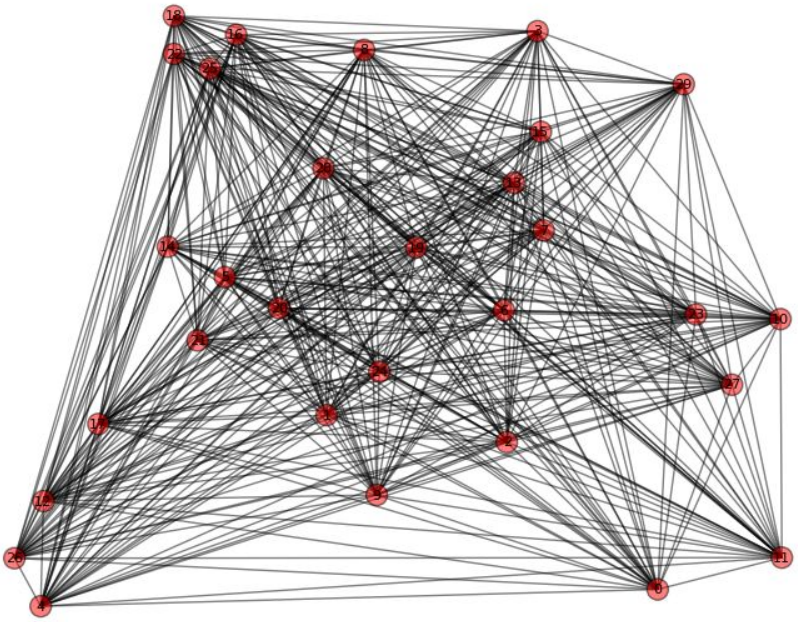
Rafael Bastos Saito - 726580
Renata Sarmet Smiderle Mendes - 726586
Rodrigo Pesse de Abreu - 726588

1.PROJETO 2: ÁRVORE GERADORA MÍNIMA

Projeto sugerido pelo professor:

A partir de um dataset específico (grafo ponderado armazenado em arquivo .gml, .graphml, .txt, .net, etc) implementar o algoritmo de Prim para extrair uma Minimum Spanning Tree (MST) de G.



Grafo apresentado - 30 cidades do mundo

A partir do pseudocódigo apresentado em sala, o algoritmo de Prim foi desenvolvido pelo grupo, resultando em:

```
4   import networkx as nx
5   import numpy as np
6   import matplotlib.pyplot as plt
7
8   toggle = True
9   G = nx.Graph()
10  H = {}
11  j = 0
12
13  def Prim(G = nx.Graph(), R = None):
14      i = 0
15      FilaPrioridade  = {}
16      predecessor = {}
17
18      for v,data in G.nodes(data=True):
19          FilaPrioridade[v] = np.inf
20          predecessor[v] = 'null'
21
22      FilaPrioridade[R] = 0.0
23      MST  = nx.create_empty_copy(G)
24
25      while FilaPrioridade:
26          u = min(FilaPrioridade,key = FilaPrioridade.get)
27          del FilaPrioridade[u]
28
29          for Vizinho in G[u]:
30              if Vizinho in FilaPrioridade:
31                  if G[u][Vizinho]['weight'] < FilaPrioridade[Vizinho]:
32                      predecessor[Vizinho] = u
33                      FilaPrioridade[Vizinho] = G[u][Vizinho]['weight']
34
35          if predecessor[u] is not 'null':
36              for v1,v2,data in G.edges(data=True):
37                  if (v1 == predecessor[u] and v2 == u):
38                      MST.add_edge(v1,v2, weight=data['weight'])
39                      H[i] = MST.copy()
40                      i = i + 1
41                  elif (v1 == u and v2 == predecessor[u]):
42                      MST.add_edge(v2,v1, weight=data['weight'])
43                      H[i] = MST.copy()
44                      i = i + 1
45
```

Algoritmo de Prim

O grupo desenvolveu uma forma, que, a partir de clicks, o grafo vá se formando, com seus pesos, vértices e arestas:

```
def onclick(event):
    global toggle
    global j

    event.canvas.figure.clear()

    if toggle:
        labels = {}
        nx.draw(G, pos, with_labels=True)
        nx.draw_networkx_edge_labels(G, pos, labels)
        toggle = not toggle

    else:
        labels = {}
    for v1,v2,data in H[j].edges(data=True):
            labels[(v1,v2)] = data['weight']
        nx.draw(H[j], pos, with_labels=True)
        nx.draw_networkx_edge_labels(H[j], pos, labels)
        j = j + 1

    event.canvas.draw()

A = np.loadtxt('ha30_dist.txt')
G = nx.from_numpy_matrix(A)
nx.draw_networkx(G)
plt.savefig('Inicial.pdf')
plt.show()

Prim(G, 0)

pos = nx.spring_layout(G)
fig = plt.figure()
fig.canvas.mpl_connect('button_press_event', onclick)
```

Algoritmo de plotagem a partir de clicks

Resultado:

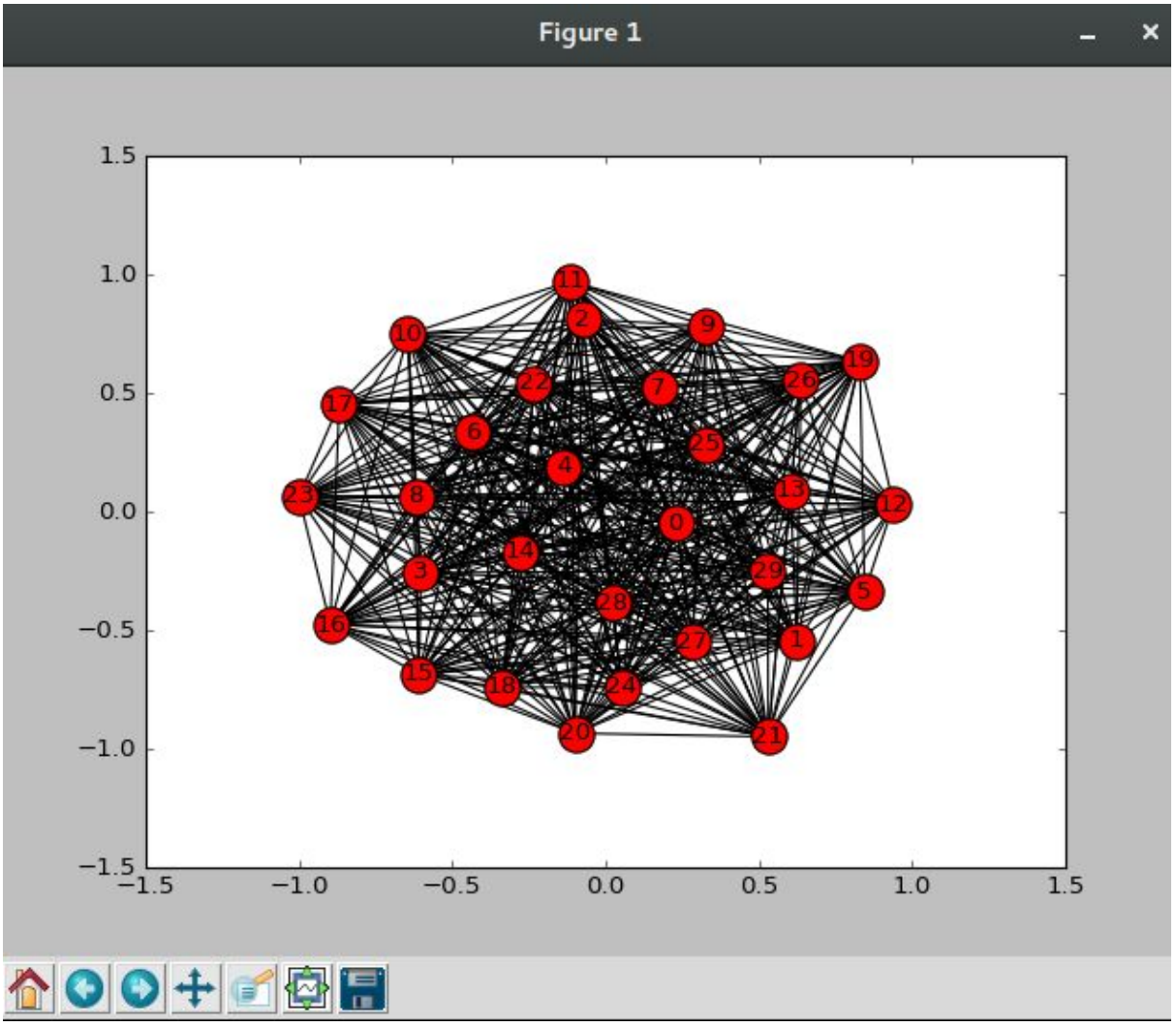
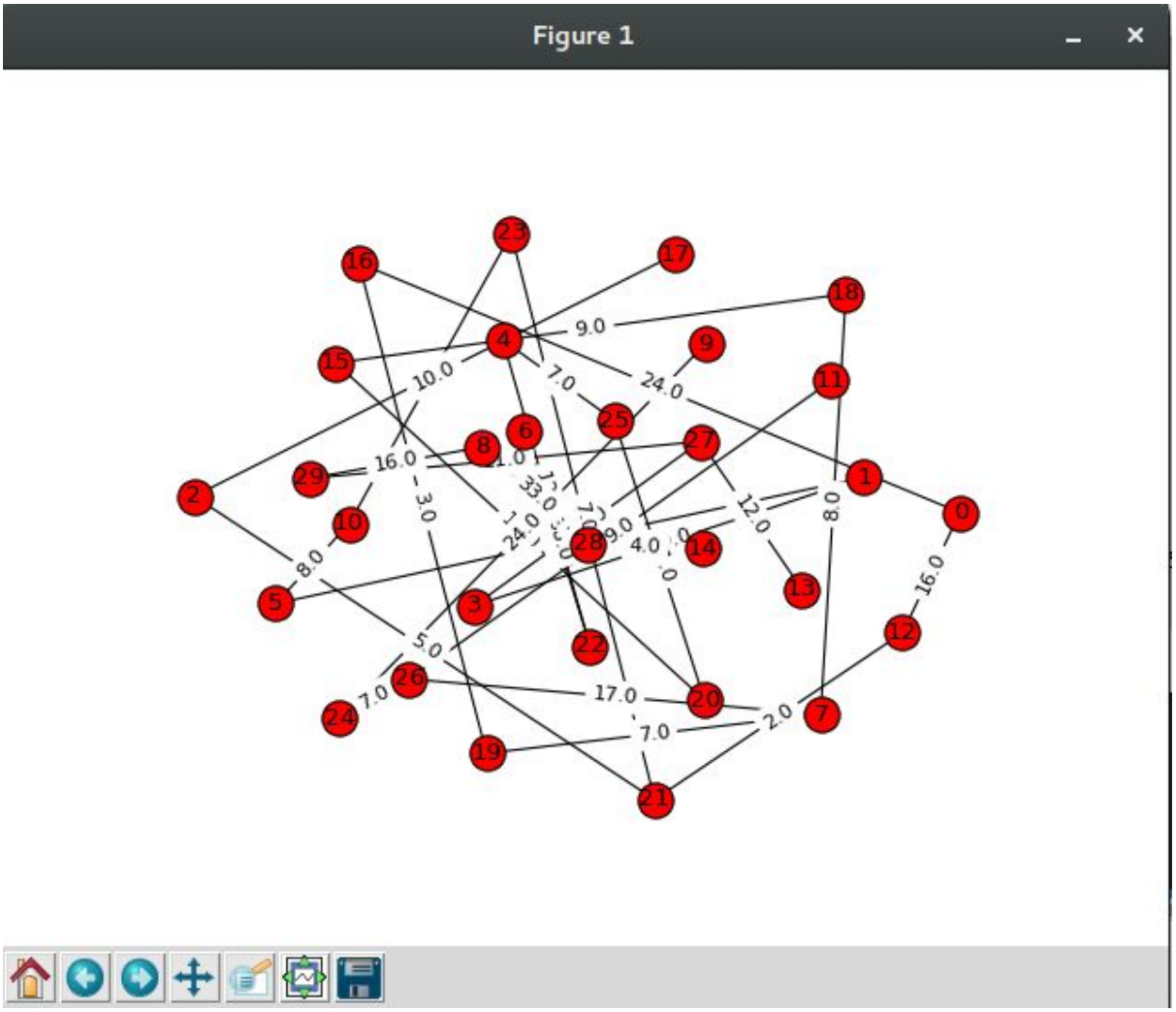


Imagem inicial



Resultado a partir da aplicação do algoritmo