Geographic Information Systems

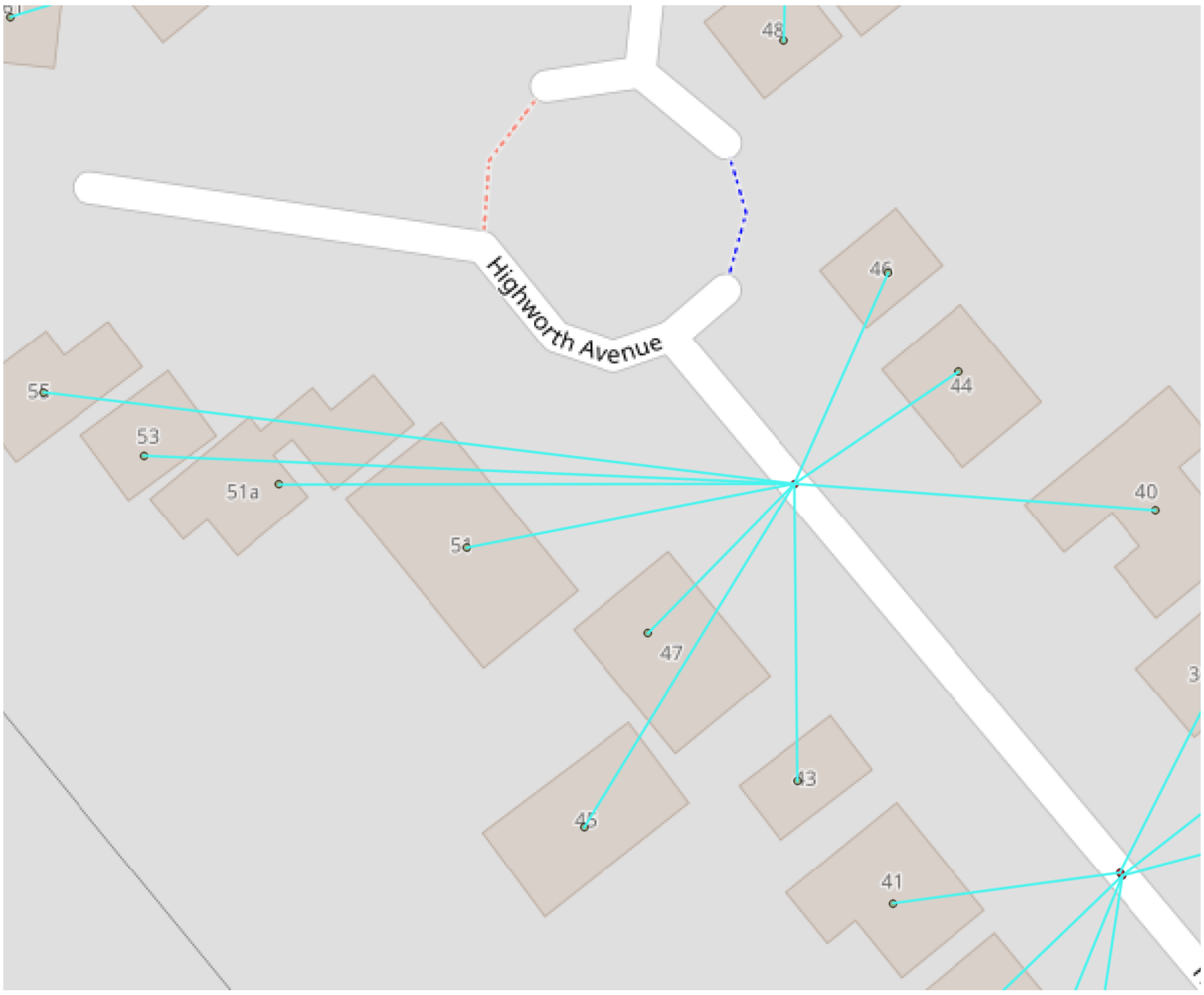
Generating Steiner tree using Network X in Python?

Asked 2 years, 9 months ago Active 2 years, 9 months ago Viewed 3k times

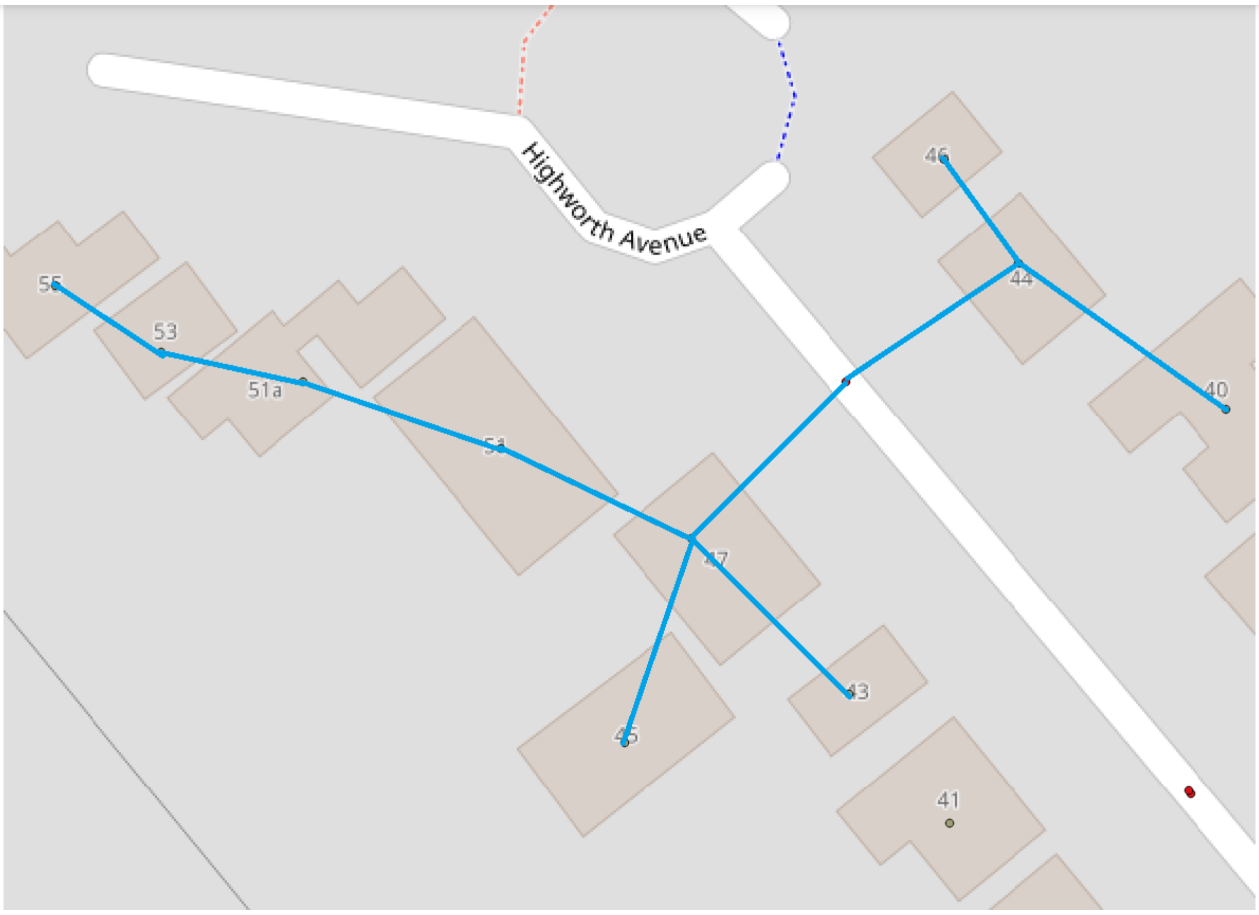
3

3

I'm trying to generate a Steiner tree in Python using NetworkX. I have a range of geolocated nodes saved in the GeoJSON format, allowing me to visually validate the outputs in QGIS. A radially deployed straight-line network looks as follows:



This isn't very realistic however for a broadband network. A Steiner tree would reduce the cost for connecting all premises. Here is my artistic attempt for how I imagine it should look:



So far I've setup a function which loads in each GeoJSON and measures the distance from the central node for the edge weight:

```python
import networkx as nx
import matplotlib.pyplot as plt
from shapely.geometry import Point

def load_nodes(premises_data, distribution_point_data):

    G = nx.Graph()

    for dist_point in distribution_point_data:
        dist_point_name = [dist_point['properties']['name']]
        dist_point_coordinates = dist_point['geometry']['coordinates']
        G.add_nodes_from(dist_point_name , pos=dist_point_coordinates )

        for premises in premises_data:
            if dist_point['properties']['name'] == premises ['properties']['link']:
                premises_name = [premises ['properties']['name']]
                premises_coordinates = premises['geometry']['coordinates']
                G.add_nodes_from(node_name, pos=premises_coordinates )

    for dist_point in distribution_point_data:
        for premises in premises_data:
            if dist_point['properties']['name'] == premises['properties']['link']:
                dist_point_geom = Point(dist_point['geometry']['coordinates'])
                premises_geom = Point(premises['geometry']['coordinates'])
                distance = round(dist_point_geom.distance(premises_geom),2)
                G.add_edge(dist_point['properties']['name'],premises['properties']
['name'],weight=distance)

    return G
```

follows:

```python
from itertools import combinations, chain
from networkx.utils import pairwise, not_implemented_for

def metric_closure(G, weight='weight'):
    """  Return the metric closure of a graph.

    The metric closure of a graph *G* is the complete graph in which each edge
    is weighted by the shortest path distance between the nodes in *G* .

    Parameters
    ----------
    G : NetworkX graph

    Returns
    -------
    NetworkX graph
        Metric closure of the graph `G`.

    """
    M = nx.Graph()

    Gnodes = set(G)

    # check for connected graph while processing first node
    all_paths_iter = nx.all_pairs_dijkstra(G, weight=weight)
    u, (distance, path) = next(all_paths_iter)
    if Gnodes - set(distance):
        msg = "G is not a connected graph. metric_closure is not defined."
        raise nx.NetworkXError(msg)
    Gnodes.remove(u)
    for v in Gnodes:
        M.add_edge(u, v, distance=distance[v], path=path[v])

    # first node done -- now process the rest
    for u, (distance, path) in all_paths_iter:
        Gnodes.remove(u)
        for v in Gnodes:
            M.add_edge(u, v, distance=distance[v], path=path[v])

    return M

def steiner_tree(G, terminal_nodes, weight='weight'):
    """ Return an approximation to the minimum Steiner tree of a graph.

    Parameters
    ----------
    G : NetworkX graph

    terminal_nodes : list
         A list of terminal nodes for which minimum steiner tree is
         to be found.

    Returns
    -------
    NetworkX graph
        Approximation to the minimum steiner tree of `G` induced by
        `terminal_nodes` .

    Notes
    -----
    Steiner tree can be approximated by computing the minimum spanning
    tree of the subgraph of the metric closure of the graph induced by the
    terminal nodes.

    """
    # M is the subgraph of the metric closure induced by the terminal nodes of
    # G.
    M = metric_closure(G, weight=weight)
    # Use the 'distance' attribute of each edge provided by the metric closure
    # graph.
    H = M.subgraph(terminal_nodes)
    mst_edges = nx.minimum_spanning_edges(H, weight='distance', data=True)
    # Create an iterator over each edge in each shortest path; repeats are okay
    edges = chain.from_iterable(pairwise(d['path']) for u, v, d in mst_edges)
    T = G.edge_subgraph(edges)
    return T
```

Here is the GeoJSON data for a reproducible example for the premises on the far right of the pictures above:

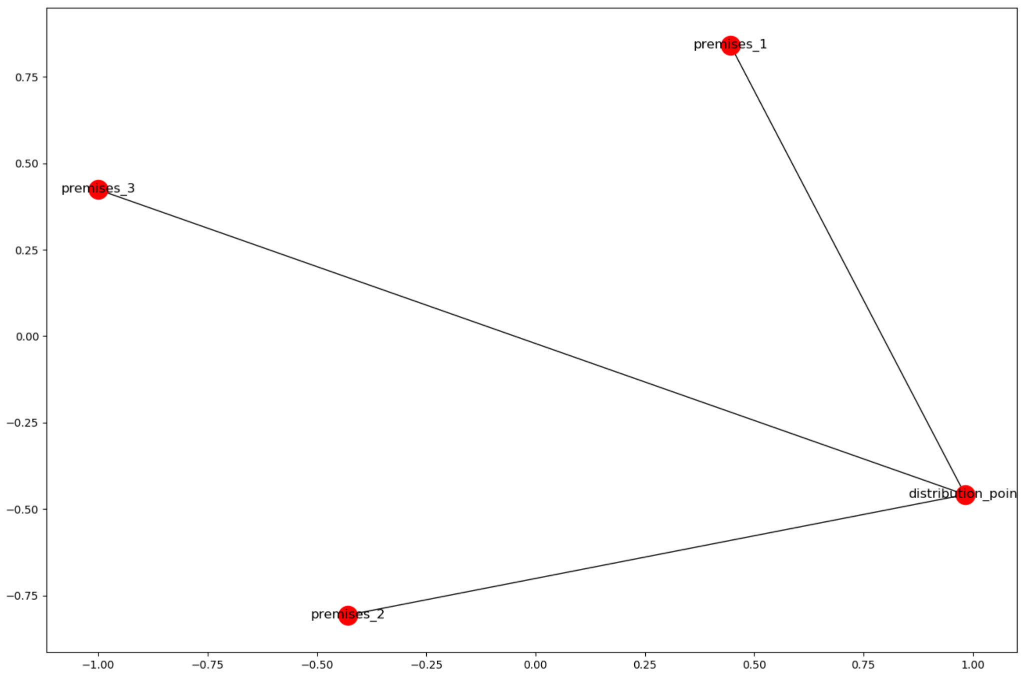
```
GEOJSON_PREMISES = [
        {
            'type': "Feature",
            'geometry':{
                "type": "Point",
                "coordinates": [0.129960,52.220977]
            },
            'properties': {
                'name': 'premises_1',
                'link': 'distribution_point'
            }
        },
        {
            'type': "Feature",
            'geometry':{
                "type": "Point",
                "coordinates": [0.130056,52.220895]
            },
            'properties': {
                'name': 'premises_2',
                'link': 'distribution_point'
            }
        },
        {
            'type': "Feature",
            'geometry':{
                "type": "Point",
                "coordinates": [0.130318,52.220782]
            },
            'properties': {
                'name': 'premises_3',
                'link': 'distribution_point'
            }
        },
    ]
GEOJSON_DISTRIBUTION_POINT = [
    {
        'type': "Feature",
        'geometry':{
            "type": "Point",
            "coordinates": [0.129829,52.220802]
        },
        'properties': {
```

And I run the following to generate a graph, and then (supposedly) a Steiner tree:

```
graph = load_nodes(GEOJSON_PREMISES, GEOJSON_DISTRIBUTION_POINT)
graph2 = steiner_tree(graph, graph.nodes, weight = 'distance')

plt.figure()
nx.draw_networkx(graph2)
plt.show()
```

This is the output I get, which was not what I was expecting:



Firstly, what am I doing wrong for generating a Steiner tree graph which allows the network to run between the premises points, like in my artistic impression?

Secondly, why does this output plot not use the geographic coordinates of the nodes? (I think I'm configuring this incorrectly because I think Network X can deal with node coordinates as well as weights).

Answers should be for Python and not QGIS/ArcMap, as this needs to eventually be run on a large-scale computer cluster.

python geojson shapely networkx

Share Edit Follow Flag edited Jan 2 '19 at 11:35 asked Jan 1 '19 at 20:19
Thirst for Knowledge

You are talking about Steiner tree, not mst – FelixIP Jan 1 '19 at 20:33

@FelixIP Question now updated to reflect this. – Thirst for Knowledge Jan 2 '19 at 11:35

Number of nodes connected by Steiner tree << total nodes in network. – FelixIP Jan 2 '19 at 19:05

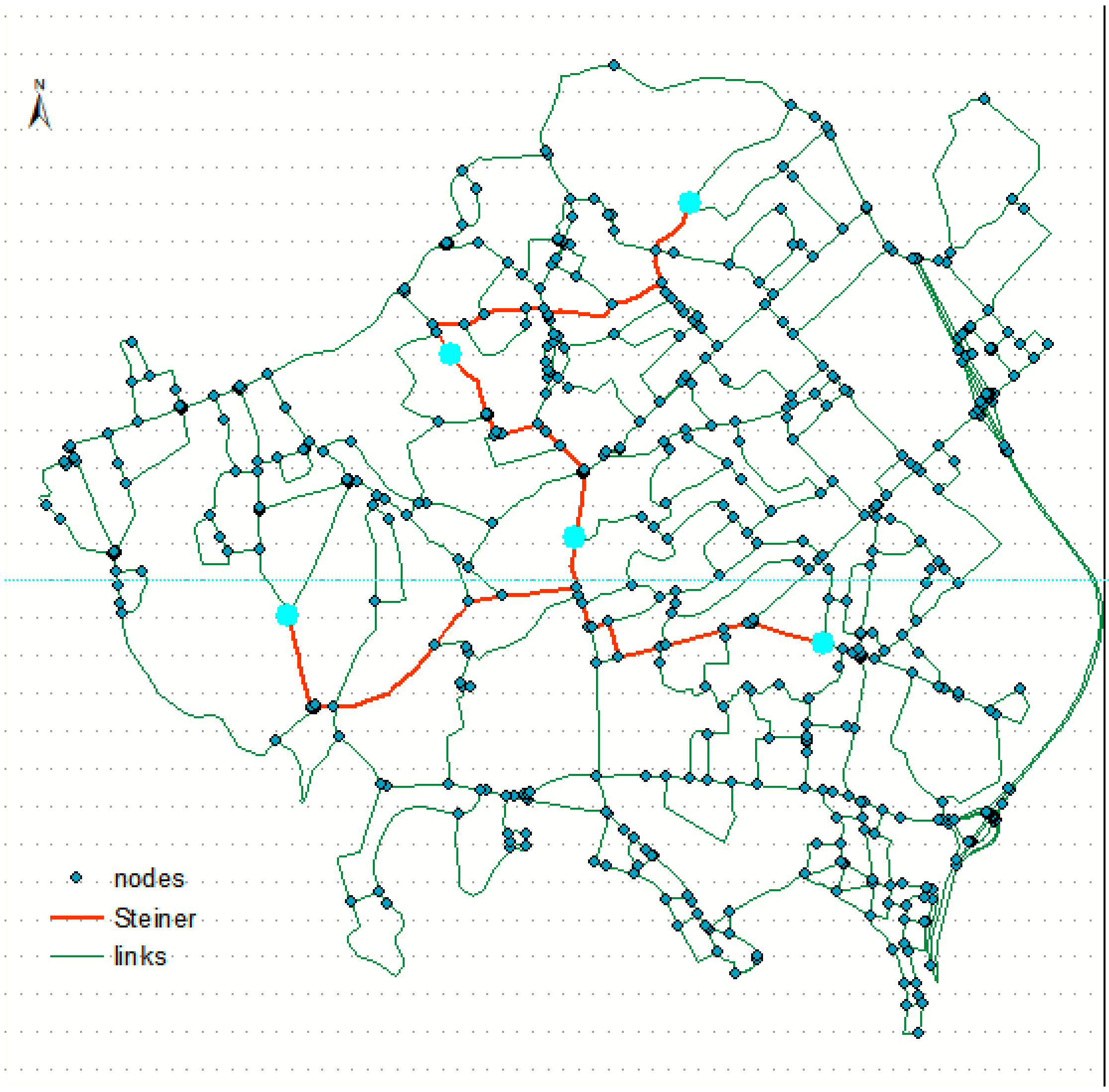
Missing node at intersection of p2-p3 and p1-source. – FelixIP Jan 2 '19 at 19:20

2 Answers

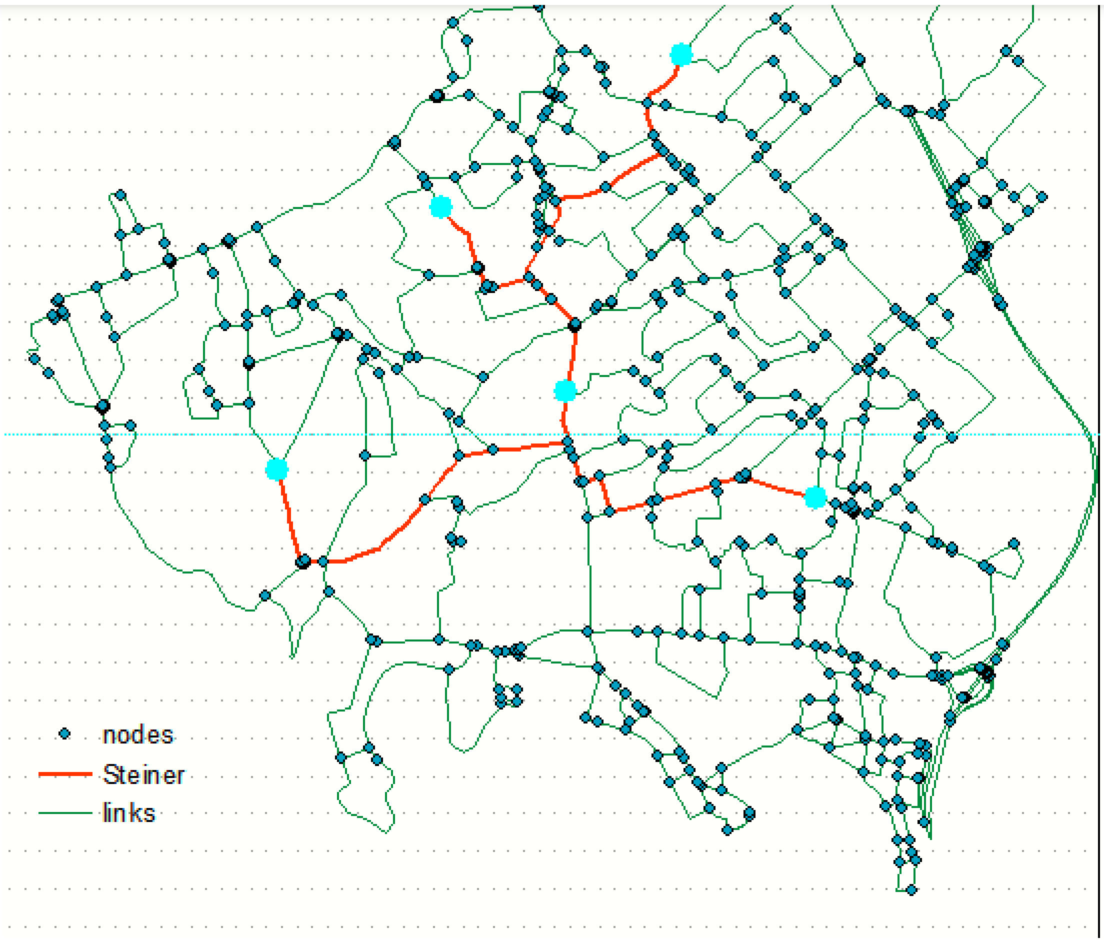
Active Oldest Votes

5

Steiner tree connects some(!) of the network's nodes (terminals) shown as selected nodes:

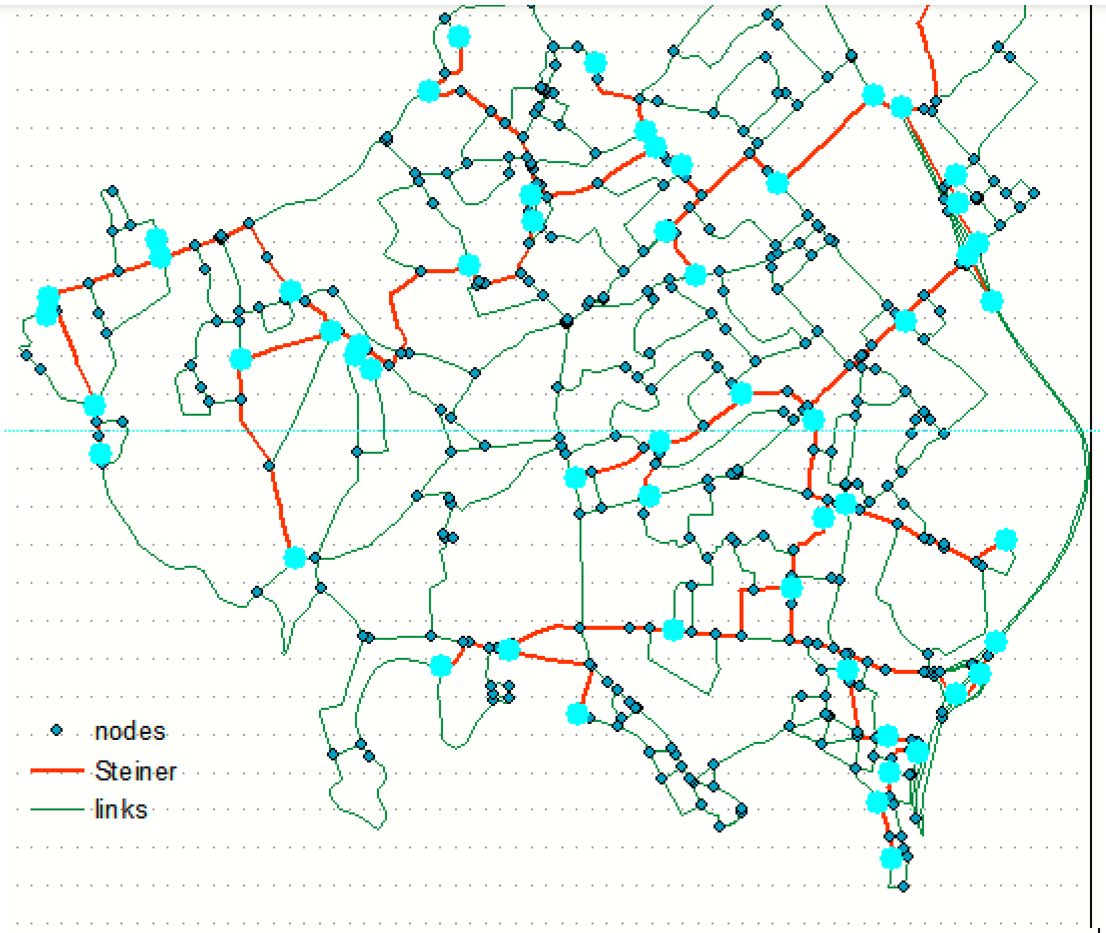


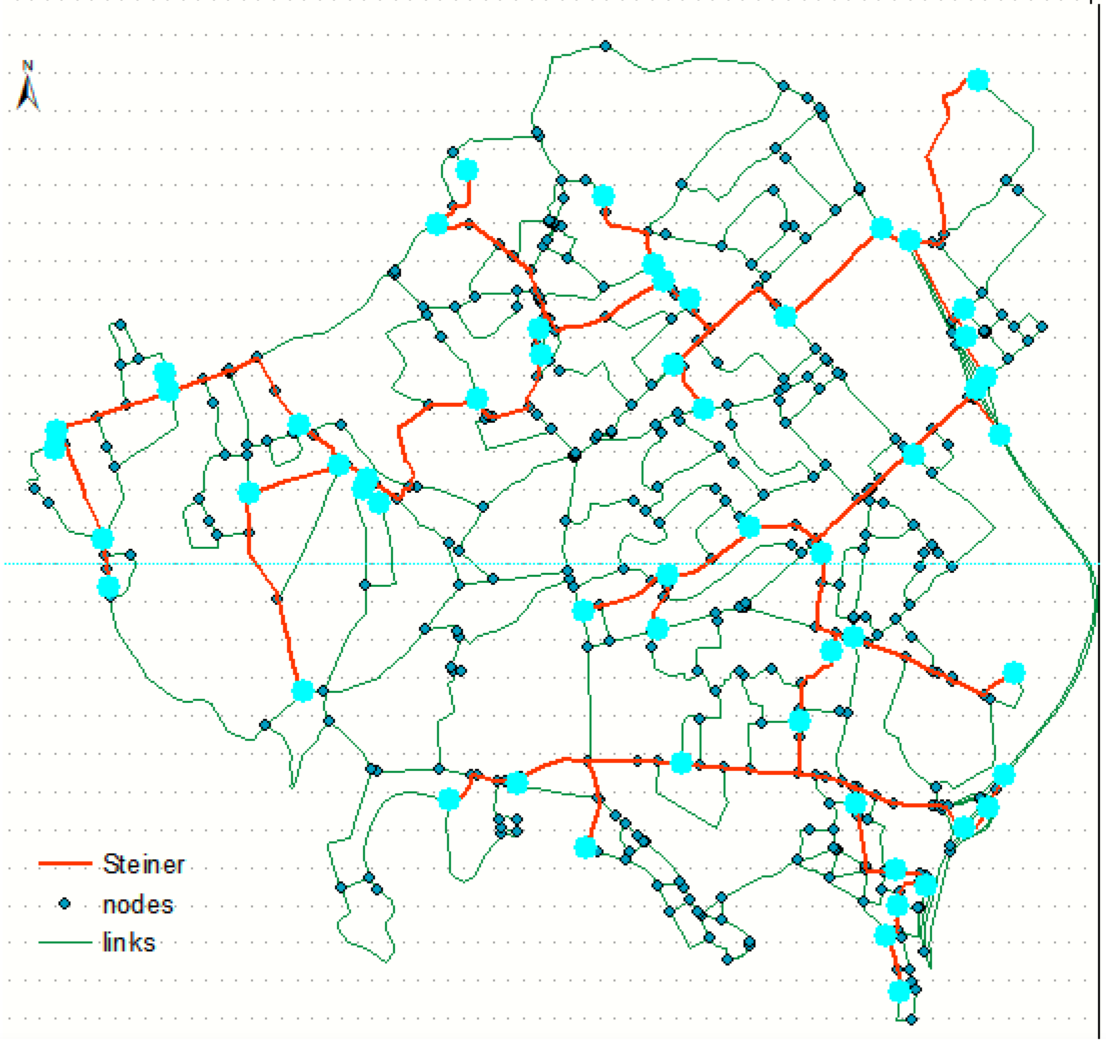
However don't get over excited about this feature of networkX, there is a good reason they called it "**approximation**.steinertree.steiner_tree", e.g. tree length in below picture is 380 m (4%) less than in the first one:



Computation of Steiner tree is ginormous task, it involves search for so called Steiners' points. Result shown on second picture is my poor attempt to improve default outcome (1st picture) by considering 3 terminals at a time in a search for such points.

NetworkX algorithm not searching for that points. It assumes that Steiner tree is minimum spanning tree of graph represented by OD matrix for terminals. That is pragmatic approach working well enough with dense terminals, where chances of Steiner points sitting on a shortest path between 2 terminals are big, e.g. difference in tree length shown on 2 pictures below is only 2%:


● nodes
Steiner
links


N
Steiner
● nodes
links

Share Edit Follow Flag edited Jan 3 '19 at 0:43 answered Jan 3 '19 at 0:38
FelixIP
20.6k 3 26 56

2

The minimum spanning tree is computed on the input graph. Your input graph is the star network from the distribution point to the three premises - it doesn't contain edges from the premises to the other premises, so the output MST can't have those links in it.

You want to create a full graph with all edges between all the pairs of locations - distribution point and premises points. Then the MST will span over that.

Share Edit Follow Flag answered Jan 1 '19 at 20:32
Spacedman
45.4k 4 53 75

I've taken on board the comment from FelixIP that this should be a Steiner Tree, and have edited the question to reflect this. In the code posted, the Steiner Tree function initially obtains the full metric closure for the graph, as you suggest. However, even after doing this I am still unable to generate the Steiner graph. Help would be much appreciated. – Thirst for Knowledge Jan 2 '19 at 11:40