This package implements community detection.

Package name is community but refer to python-louvain on pypi

```python
community.best_partition(graph, partition=None, weight='weight', resolution=1.0, randomize=None, random_state=None)
```

Compute the partition of the graph nodes which maximizes the modularity (or try..) using the Louvain heuristics

This is the partition of highest modularity, i.e. the highest partition of the dendrogram generated by the Louvain algorithm.

**Parameters**

- `graph` [networkx.Graph] the networkx graph which is decomposed
- `partition` [dict, optional] the algorithm will start using this partition of the nodes. It's a dictionary where keys are their nodes and values the communities
- `weight` [str, optional] the key in graph to use as weight. Default to 'weight'
- `resolution` [double, optional] Will change the size of the communities, default to 1. represents the time described in “Laplacian Dynamics and Multiscale Modular Structure in Networks”, R. Lambiotte, J.-C. Delvenne, M. Barahona
- `randomize` [boolean, optional] Will randomize the node evaluation order and the community evaluation order to get different partitions at each call
- `random_state` [int, RandomState instance or None, optional (default=None)] If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

**Returns**

- `partition` [dictionary] The partition, with communities numbered from 0 to number of communities

**Raises**

- `NetworkXError` If the graph is not Eulerian.

**See also:**

- `generate_dendrogram` to obtain all the decompositions levels

**Notes**

Uses Louvain algorithm

**References**


**Examples**

```python
>>> # basic usage
>>> import community as community_louvain
>>> import networkx as nx
```
>>> G = nx.erdos_renyi_graph(100, 0.01)
>>> partition = community_louvain.best_partition(G)

>>> # display a graph with its communities:
>>> # as Erdos-Renyi graphs don't have true community structure,
>>> # instead load the karate club graph
>>> import community as community_louvain
>>> import matplotlib.cm as cm
>>> import matplotlib.pyplot as plt
>>> import networkx as nx

>>> G = nx.karate_club_graph()

>>> # compute the best partition
>>> partition = community_louvain.best_partition(G)

>>> # draw the graph
>>> pos = nx.spring_layout(G)

>>> # color the nodes according to their partition
>>> cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
>>> nx.draw_networkx_nodes(G, pos, partition.keys(), node_size=40,
>>> cmap=cmap, node_color=list(partition.values()))

>>> nx.draw_networkx_edges(G, pos, alpha=0.5)
>>> plt.show()

community.generate_dendrogram(graph, part_init=None, weight='weight', resolution=1.0, randomize=None, random_state=None)

Find communities in the graph and return the associated dendrogram

A dendrogram is a tree and each level is a partition of the graph nodes. Level 0 is the first partition, which contains the smallest communities, and the best is len(dendrogram) - 1. The higher the level is, the bigger are the communities

Parameters

- **graph** [networkx.Graph] the networkx graph which will be decomposed
- **part_init** [dict, optional] the algorithm will start using this partition of the nodes. It’s a dictionary where keys are their nodes and values the communities
- **weight** [str, optional] the key in graph to use as weight. Default to ‘weight’
- **resolution** [double, optional] Will change the size of the communities, default to 1. represents the time described in “Laplacian Dynamics and Multiscale Modular Structure in Networks”, R. Lambiotte, J.-C. Delvenne, M. Barahona

Returns

- **dendrogram** [list of dictionaries] a list of partitions, ie dictionaries where keys of the i+1 are the values of the i. and where keys of the first are the nodes of graph

Raises

- **TypeError** If the graph is not a networkx.Graph

See also:

- **best_partition**
Notes

Uses Louvain algorithm

References


Examples

```python
>>> G=nx.erdos_renyi_graph(100, 0.01)
>>> dendo = generate_dendrogram(G)
>>> for level in range(len(dendo) - 1) :
...     print("partition at level", level,
...     "is", partition_at_level(dendo, level))
:param weight:
:type weight:

community.induced_graph (partition, graph, weight='weight')
Produce the graph where nodes are the communities

there is a link of weight w between communities if the sum of the weights of the links between their elements is w

Parameters

partition [dict] a dictionary where keys are graph nodes and values the part the node belongs to

graph [networkx.Graph] the initial graph

weight [str, optional] the key in graph to use as weight. Default to ‘weight’

Returns

: g [networkx.Graph] a networkx graph where nodes are the parts

Examples

```python
>>> n = 5
>>> g = nx.complete_graph(2*n)
>>> part = dict()
>>> for node in g.nodes() :
...     part[node] = node % 2
>>> ind = induced_graph(part, g)
>>> goal = nx.Graph()
>>> goal.add_weighted_edges_from([(0,1,n*n),(0,0,n*(n-1)/2), (1, 1, n*(n-1)/2)])
>>> nx.is_isomorphic(ind, goal)
True
```

community.load_binary (data)
Load binary graph as used by the cpp implementation of this algorithm

community.modularity (partition, graph, weight='weight')
Compute the modularity of a partition of a graph
Parameters

- **partition** [dict] the partition of the nodes, i.e a dictionary where keys are their nodes and values the communities
- **graph** [networkx.Graph] the networkx graph which is decomposed
- **weight** [str, optional] the key in graph to use as weight. Default to ‘weight’

Returns

- **modularity** [float] The modularity

Raises

- **KeyError** If the partition is not a partition of all graph nodes
- **ValueError** If the graph has no link
- **TypeError** If graph is not a networkx.Graph

References


Examples

```python
>>> import community as community_louvain
>>> import networkx as nx
>>> G = nx.erdos_renyi_graph(100, 0.01)
>>> partition = community_louvain.best_partition(G)
>>> modularity(partition, G)
```

Community detection for NetworkX Documentation, Release 2

```python
community.partition_at_level(dendrogram, level)
```

Return the partition of the nodes at the given level

A dendrogram is a tree and each level is a partition of the graph nodes. Level 0 is the first partition, which contains the smallest communities, and the best is len(dendrogram) - 1. The higher the level is, the bigger are the communities

Parameters

- **dendrogram** [list of dict] a list of partitions, ie dictionnaries where keys of the i+1 are the values of the i.
- **level** [int] the level which belongs to [0..len(dendrogram)-1]

Returns

- **partition** [dictionary] A dictionary where keys are the nodes and the values are the set it belongs to

Raises

- **KeyError** If the dendrogram is not well formed or the level is too high

See also:

- **best_partition** which directly combines partition_at_level and
- **generate_dendrogram** to obtain the partition of highest modularity
Examples

```python
>>> G = nx.erdos_renyi_graph(100, 0.01)
>>> dendrogram = generate_dendrogram(G)
>>> for level in range(len(dendrogram) - 1):
...     print("partition at level", level, "is", partition_at_level(dendrogram,
...                                                            level))  # NOQA
```
CHAPTER 1

Indices and tables

- genindex
- modindex
- search
Community detection for NetworkX’s documentation

This module implements community detection.

It uses the louvain method described in Fast unfolding of communities in large networks, Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Renaud Lefebvre, Journal of Statistical Mechanics: Theory and Experiment 2008(10), P10008 (12pp)

It depends on Networkx to handle graph operations: [http://networkx.lanl.gov/](http://networkx.lanl.gov/)

The program can be found in a repository where you can also report bugs:
CHAPTER 3

Example:

3.1 As a command line utility:

You should consider using the cpp version at http://findcommunities.googlepages.com/!

`./community.py file.bin > tree`

where file.bin is a binary graph as generated by the convert utility of the cpp version.
The generated file can then be used with the hierarchy utility of the cpp version. Note that the program does not make many verifications about the arguments, and is expecting a friendly use.

3.2 As python module:

```python
import community as community_louvain
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import networkx as nx

# load the karate club graph
G = nx.karate_club_graph()

# first compute the best partition
partition = community_louvain.best_partition(G)

# color the nodes according to their partition

cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
nx.draw_networkx_nodes(G, pos, partition.keys(), node_size=40,
```
cmap=cmap, node_color=list(partition.values()))
nx.draw_networkx_edges(G, pos, alpha=0.5)
plt.show()
Changelog:

• 2020-04-06 : 0.14, bugfixes (on resolution parameter), optimization on random state
• 2018-12-21 : 0.13, better random state, some files missing included, communities always in 0..N-1
• 2018-05-22 : 0.11, stop forcing networkx<2.0 and expose module __version__
• 2018-01-02 : 0.10, bug fix: taking into account the node removal cost
• 2017-09-21 : 0.9, support networkx 2.0
• 2017-06-03 : 0.8, add randomization and bugfixes
• 2017-05-21 : 0.7, migrate to github, readthedocs and travis. Add resolution parameter to control community size, bugfixes
• 04/21/2011 : modifications to use networkx like documentation and use of test.
• 02/22/2011 : correction of a bug regarding edge weights
• 01/14/2010 : modification to use networkx 1.01 graph api and adding the possibility to start the algorithm with a given partition
• 04/10/2009 : increase of the speed of the detection by caching node degrees
CHAPTER 5

License:
CHAPTER 6

Indices and tables

- genindex
- modindex
- search
Python Module Index

C

community, 1
Index

B
best_partition() *(in module community)*, 1

C
community *(module)*, 1

G
generate_dendrogram() *(in module community)*, 2

I
induced_graph() *(in module community)*, 3

L
load_binary() *(in module community)*, 3

M
modularity() *(in module community)*, 3

P
partition_at_level() *(in module community)*, 4