Community detection algorithms in Python

NetworkX

Community detection algorithms covered in lecture available in NetworkX. Additional algorithms can be found here: https://networkx.org/documentation/networkx-2.5/reference/algorithms/community.html

Girvan-Newman Algorithm

- **docs**: https://networkx.org/documentation/networkx-2.5/reference/algorithms/generated/networkx.algorithms.community.centrality.girvan_newman.html#networkx.algorithms.community.centrality.girvan_newman
- *Girvan_newman(G, most_valuable_edge=None)*
  - *most_valuable_edge* can be used to specify a function that takes the graph as an input and outputs an edge. If no function is provided the edge with highest *edge_betweenness_centrality()* will be used.
- Can be used for **weighted** graphs, directed graphs are treated as an undirected graph.
  - By default weights are not taken into account, but can be included by a user defined function given to *most_valuable_edge*
  - Can take directed or undirected graphs, but algorithm always treats graph as undirected

```
from networkx.algorithms.community.centrality import girvan_newman

G = nx.karate_club_graph()
partitions = girvan_newman(G)
```

Fast Greedy Modularity

- **docs**: https://networkx.org/documentation/networkx-2.5/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html#networkx.algorithms.community.modularity_max.greedy_modularity_communities
- *greedy_modularity_communities(G, weight=None)*
- Can be used for **weighted** graphs, expects an undirected graph.

```
from networkx.algorithms.community import greedy_modularity_communities

G = nx.karate_club_graph()
partitions = greedy_modularity_communities(G)
```

CDLIB

CDlib is a python package that provides community detection algorithms to be used for both NetworkX or igraph networks. The library can be installed using pip.

```
pip install igraph
pip install cdlib
```

Random Walk: Pons & Latapy

- **docs**: https://cdlib.readthedocs.io/en/latest/reference/cd_algorithms/algs/cdlib.algorithms.walktrap.html#cdlib.algorithms.walktrap
- *walktrap(G)*
- Does not take weights into account and directed graphs are considered undirected.
  - algorithm converts networkx graph to igraph

```
from cdlib import algorithms

G = nx.karate_club_graph()
partitions = algorithms.walktrap(G)
```

Leading eigenvector

- **docs**: https://cdlib.readthedocs.io/en/latest/reference/cd_algorithms/algs/cdlib.algorithms.eigenvector.html?highlight=leading%20eigenvector
- *eigenvector(G)*
- Does not take weights into account and directed graphs are considered undirected
  - algorithm converts networkx graph to igraph

```
from cdlib import algorithms

G = nx.karate_club_graph()
partitions = algorithms.eigenvector(G)
```

Louvain

- **docs**: https://cdlib.readthedocs.io/en/latest/reference/cd_algorithms/algs/cdlib.algorithms.louvain.html?highlight=louvain
- *louvain(G, weight='weight', resolution=1.0, randomize=False)*
- Can be used for **weighted** graphs, but not directed graphs
  - algorithm converts networkx graph to igraph

```
from cdlib import algorithms

G = nx.karate_club_graph()
partitions = algorithms.louvain(G)
```

Infomap

- **docs**: https://cdlib.readthedocs.io/en/latest/reference/cd_algorithms/algs/cdlib.algorithms.infomap.html?highlight=infomap
- *infomap(G)*
- Can be used for **weighted** graphs and **directed** graphs

```
from cdlib import algorithms

G = nx.karate_club_graph()
partitions = algorithms.louvain(G)
```