Graph

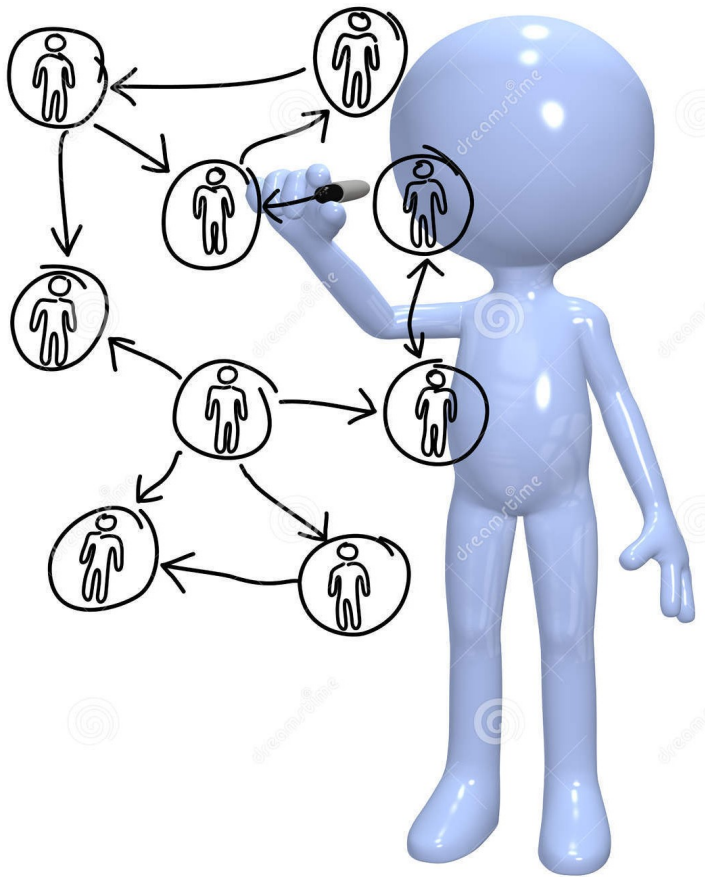
Vladimir Batagelj

Inštitut za matematiko, fiziko in mehaniko
Oddelek za teoretično računalništvo

Manual

Ljubljana, August 2016

Outline



Last version (September 9, 2016 at 13 : 20):

Vladimir Batagelj: vladimir.batagelj@fmf.uni-lj.si

Networks

A **graph** $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ consists of the set of nodes $\mathcal{V}$ and the set of links $\mathcal{L}$. A link is either directed, an arc, or undirected, an edge – $\mathcal{L} = \mathcal{A} \cup \mathcal{E}$, $\mathcal{A} \cup \mathcal{E} = \emptyset$ where $\mathcal{A}$ is the set of arcs and $\mathcal{E}$ is the set of edges.

A **network** $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W})$ – is a graph with node properties $\mathcal{P}$ an link properties or weights $\mathcal{W}$.

In a **two-mode network** $\mathcal{N} = ((\mathcal{V}_1, \mathcal{V}_2), \mathcal{L}, \mathcal{P}, \mathcal{W})$ – the set of nodes is split into two disjoint subsets. Each link has an end-node in each subset.

In a **multirelational network** $\mathcal{N} = (\mathcal{V}, (\mathcal{L}_i, i \in I), \mathcal{P}, \mathcal{W})$ – the set of links is partitioned to several subsets – relations
(Subject Verb Object).

In a **temporal network** $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W}, \mathcal{T})$ – the time component $\mathcal{T}$ is added. To each node and link its activity set (of time points) is assigned. Also properties of nodes and links can change through time – temporal quantities.

A **collection** of networks – networks with common subsets of nodes.

The types can be combined. For example: two-mode multirelational temporal network.

Data structure

Every node/link has an id. For links, if not provided by the user, it is assigned by the package.
The data structure `graph` is composed from 3 dictionaries:

- `graph` – keys: properties of the network. Some properties are fixed: `simple`, `directed`, `multirel`, `mode`, `temporal`. The user can add other properties – for example: `nNodes`, `nArcs`, `nWeak`, `planar`, `maxT`, etc.
- `nodes` – keys are node ids. The value is a list of four dictionaries:
  [edgeStar, inArcStar, outArcStar, nodeProperties]
  Each star has node ids as keys with a list of link ids as value.
- `links` – keys are link ids. The value is a list
  [nodeId1, nodeId2, directed, relId, linkProperties]
  where linkProperties is again a dictionary.

Functions

Work in progress !!!

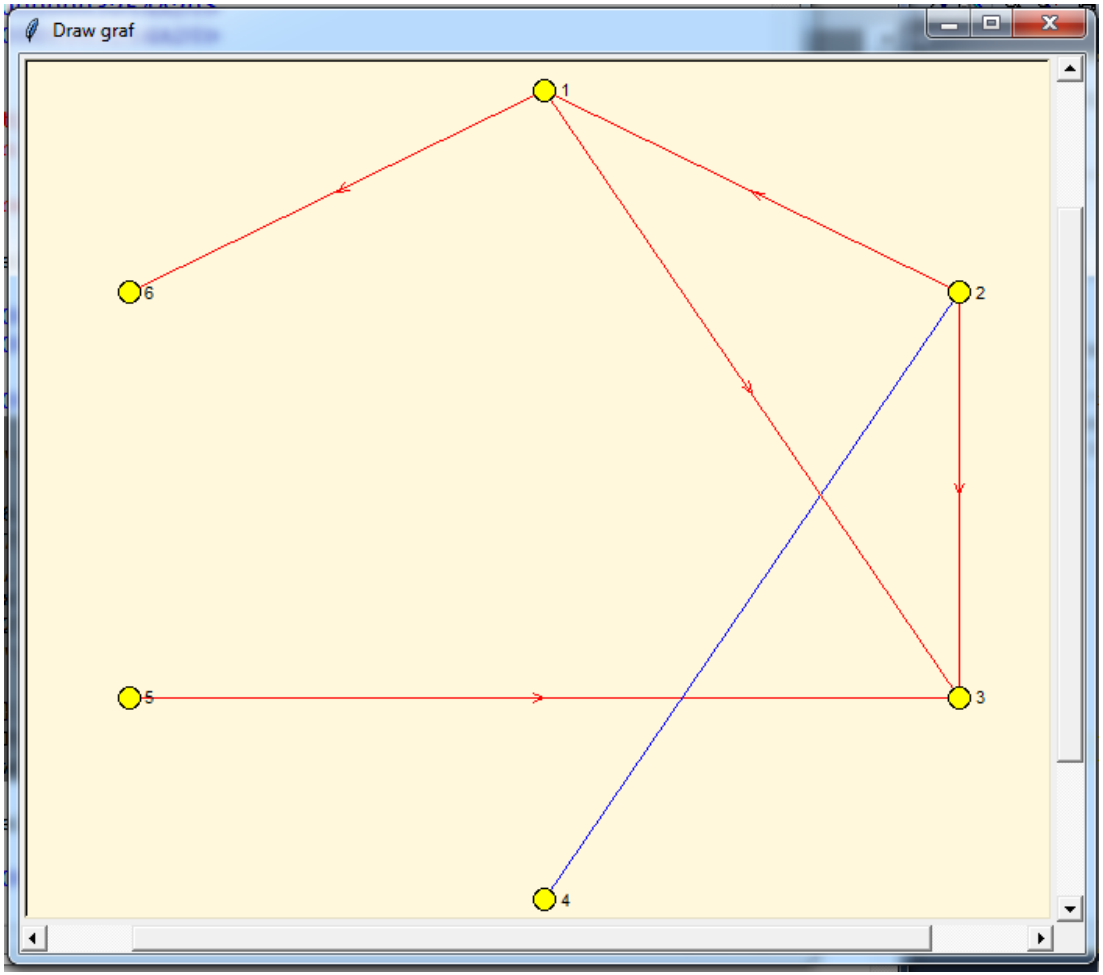
See the code.

In the version `GraphNew.py` a new implementation of multiple links between a pair of nodes was done. Not all other functions were tested yet.

Test of graph constructors

```
from GraphNew import Graph
def TestAdd():
    G = Graph()
    G.addNode(2); G.addNode(1); G.addNode(3); G.addNode(4)
    G.addEdge(2,4,{'w':3}); G.addArc(2,1,{'w':5});
    G.addArc(1,3,{'w':4}); G.addArc(2,3,{'w':6})
    G.addNode(5); G.addNode(6)
    i=G.addArc(5,3,{'w':5}); j=G.addEdge(2,4,{'w':7});
    G.addArc(1,6,{'w':8});G.addArc(1,3,{'w':5})
    G.onCircle()
    print(G)
    G.draw(800,800,"Cornsilk")
    G.savePajek('test.net')
    G.delLink(j); G.delLink(i)
    print(G)
    return G
```

Network picture



Network data

```
>>> G._graph
{'mode': 1, 'multirel': False, 'temporal': False, 'simple': False}
>>> G._nodes
{
1: [{}, {2: [2]}, {3: [3, 8], 6: [7]}, {'x': 0.5, 'y': 0.95}],
2: [{4: [1]}, {}, {1: [2], 3: [4]}, {'x': 0.88971, 'y': 0.725}],
3: [{}, {1: [3, 8], 2: [4], 5: []}, {}, {'x': 0.88971, 'y': 0.275}],
4: [{2: [1]}, {}, {}, {'x': 0.5, 'y': 0.045}],
5: [{}, {}, {3: []}, {'x': 0.11029, 'y': 0.275}],
6: [{}, {1: [7]}, {}, {'x': 0.11029, 'y': 0.725}]
}
>>> G._links
{
1: [2, 4, False, None, {'w': 3}],
2: [2, 1, True, None, {'w': 5}],
3: [1, 3, True, None, {'w': 4}],
4: [2, 3, True, None, {'w': 6}],
7: [1, 6, True, None, {'w': 8}],
8: [1, 3, True, None, {'w': 5}]
}
```

netJSON format

```
{
"netJSON": "basic",
"info": {
   "org":1, "nNodes":n, "nArcs":mA, "nEdges":mE,
   "network":label, "title":title,
   "meta": [events], ...
   },
"nodes": {
   ["id":nodeId, "lab":label, ... ],
   ***
   }
"links": {
   ["type":arc/edge, "n1":nodeID1, "n2":nodeID2, ...]
   ***
   }
}
```

... user defined properties

*** sequence of such elements

Transforming Pajek files into netJSON

```
gdir = 'c:/users/batagelj/work/python/graph/graph'
wdir = 'c:/users/batagelj/work/python/graph/JSON'
# indent = None
indent = 3
import sys, os, datetime, json
sys.path = [gdir]+sys.path; os.chdir(wdir)
import GraphNew as Graph
file='violenceM.net'
P = Graph.Graph.loadPajek(file)
# info
n=len(P); mE = len(list(P.edges())); mA = len(list(P.arcs()))
ctime=datetime.datetime.now().ctime()
title="Franzosi's violence network"
meta=[{"date":ctime, "author": "Pajek2JSON"}]
meta.append(P.getGraph('meta'))
info = {"network": "violenceM", "org": 1, "nNodes": n,
  "nArcs": mA, "nEdges": mE, "title": title, "meta": meta}
# nodes
nodes = []
for node in P.nodes():
  Node = {"id": node, "lab": P.getNode(node,"lab"),
    "tq": P.getNode(node,"tq")}
  nodes.append(Node)
```

. . . Transforming Pajek files into netJSON

```
# links
links = []
for e in P.links():
  link = P.link(e); ltype = "arc" if link[2] else "edge"
  Link = {"type": ltype, "n1": link[0], "n2": link[1],
    "rel": link[3], "tq": P.getLink(e,'tq')}
  links.append(Link)
# JSON
net = {"netJSON": "basic", "info": info, "nodes": nodes, "links": links}
js = open(info['network']+'.json','w')
json.dump(net, js, ensure_ascii=False, indent=indent)
js.close()
```