Introduction to Network Analysis using Pajek

6. Structure of networks 4
Acyclic networks and Patterns search

Vladimir Batagelj

IMFM Ljubljana and IAM Koper

Phd program on Statistics
University of Ljubljana, 2018
Outline

1. Acyclic networks
2. Numberings
3. Citation networks
4. Genealogies
5. Pattern searching
6. Triads

e-mail: vladimir.batagelj@fmf.uni-lj.si
version: May 8, 2018
Acyclic networks

Network $\mathcal{G} = (\mathcal{V}, R)$, $R \subseteq \mathcal{V} \times \mathcal{V}$ is acyclic, if it doesn’t contain any (proper) cycle.

\[ \overline{R} \cap I = \emptyset \]

In some cases we allow loops. Examples: citation networks, genealogies, project networks, . . .

In real-life acyclic networks we usually have a node property $p : \mathcal{V} \to \mathbb{R}$ (most often time), that is compatible with arcs

\[ (u, v) \in R \Rightarrow p(u) < p(v) \]

acyclic.paj
Basic properties of acyclic networks

Let $G = (\mathcal{V}, R)$ be acyclic and $\mathcal{U} \subseteq \mathcal{V}$, then $G|\mathcal{U} = (\mathcal{U}, R|\mathcal{U})$, $R|\mathcal{U} = R \cap \mathcal{U} \times \mathcal{U}$ is also acyclic.

Let $G = (\mathcal{V}, R)$ be acyclic, then $G' = (\mathcal{V}, R^{-1})$ is also acyclic.

Duality.

The set of sources $\text{Min}_R(\mathcal{V}) = \{v : \neg \exists u \in \mathcal{V} : (u, v) \in R\}$ and the set of sinks $\text{Max}_R(\mathcal{V}) = \{v : \neg \exists u \in \mathcal{V} : (v, u) \in R\}$ are nonempty (in finite networks).

Transitive closure $\overline{R}$ of an acyclic relation $R$ is acyclic.

Relation $Q$ is a skeleton of relation $R$ iff $Q \subseteq R$, $\overline{Q} = \overline{R}$ and relation $Q$ is minimal such relation – no arc can be deleted from it without destroying the second property.

A general relation (graph) can have several skeletons; but in a case of acyclic relation it is uniquely determined $Q = R \setminus R \ast \overline{R}$. 
Mapping $h : \mathcal{V} \rightarrow \mathbb{N}^+$ is called depth or level if all differences on the longest path and the initial value equal to 1.

$\mathcal{U} \leftarrow \mathcal{V}; \; k \leftarrow 0$

while $\mathcal{U} \neq \emptyset$ do

$\mathcal{T} \leftarrow \text{Min}_R(\mathcal{U}); \; k \leftarrow k + 1$

for $v \in \mathcal{T}$ do $h(v) \leftarrow k$

$\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{T}$

Drawing on levels. Macro Layers.
Acyclic
V. Batagelj
Acyclic networks
Numberings
Citation networks
Genealogies
Pattern searching
Triads

p-graph of Bouchard’s genealogy
Injective mapping $h : \mathcal{V} \rightarrow 1..|\mathcal{V}|$ compatible with relation $R$ is called a topological numbering.

'Topological sort'

\[ \mathcal{U} \leftarrow \mathcal{V}; \; k \leftarrow 0 \]

\[ \text{while } \mathcal{U} \neq \emptyset \text{ do} \]

select $v \in \text{Min}_R(\mathcal{U})$; $k \leftarrow k + 1$

$h(v) \leftarrow k$

$\mathcal{U} \leftarrow \mathcal{U} \setminus \{v\}$

Matrix display of acyclic network with vertices reordered according to a topological numbering has a zero lower triangle.
...Topological numberings

read or select  [Acyclic.paj]
Network/Acyclic Network/Depth Partition/Acyclic
Partition/Make Permutation
File/Network/Export as Matrix to EPS/Using Permutation [acy.eps]
Let the function $f : \mathcal{V} \rightarrow \mathbb{R}$ be defined in the following way:

- $f(\nu)$ is known in sources $\nu \in \text{Min}_R(\mathcal{V})$
- $f(\nu) = F(\{f(u) : uR\nu\})$

If we compute the values of function $f$ in a sequence determined by a topological numbering we can compute them in one pass since for each node $\nu \in \mathcal{V}$ the values of $f$ needed for its computation are already known.
Topological numberings – CPM

CPM (Critical Path Method): A project consists of tasks. Nodes of a project network represent states of the project and arcs represent tasks. Every project network is acyclic. For each task \((u, v)\) its execution time \(t(u, v)\) is known. A task can start only when all the preceding tasks are finished. We want to know what is the shortest time in which the project can be completed.

Let \(T(v)\) denotes the earliest time of completion of all tasks entering the state \(v\).

\[
T(v) = 0, \quad v \in \text{Min}_R(V)
\]

\[
T(v) = \max_{u:uRv} (T(u) + t(u, v))
\]

Network/Acyclic Network/Critical Path Method–CPM
The citation network analysis started in 1964 with the paper of Garfield et al. In 1989 Hummon and Doreian proposed three indices – weights of arcs that provide us with automatic way to identify the (most) important part of the citation network. For two of these indices we developed algorithms to efficiently compute them.
... Citation networks

In a given set of units/nodes \( U \) (articles, books, works, etc.) we introduce a *citing relation*/set of arcs \( R \subseteq U \times U \)

\[ uRv \equiv u \text{ cites } v \]

which determines a *citation network* \( \mathcal{N} = (U, R) \).

A citing relation is usually *irreflexive* (no loops) and (almost) *acyclic*. We shall assume that it has these two properties. Since in real-life citation networks the strong components are small (usually 2 or 3 nodes) we can transform such network into an acyclic network by shrinking strong components and deleting loops. Other approaches exist. It is also useful to transform a citation network to its *standardized* form by adding a common *source* node \( s \notin U \) and a common *sink* node \( t \notin U \). The source \( s \) is linked by an arc to all minimal elements of \( R \); and all maximal elements of \( R \) are linked to the sink \( t \). We add also the ‘feedback’ arc \((t, s)\).
The search path count (SPC) method is based on counters \( n(u, v) \) that count the number of different paths from \( s \) to \( t \) through the arc \((u, v)\). To compute \( n(u, v) \) we introduce two auxiliary quantities: \( n^-(v) \) counts the number of different paths from \( s \) to \( v \), and \( n^+(v) \) counts the number of different paths from \( v \) to \( t \).
Fast algorithm for SPC

It follows by basic principles of combinatorics that

\[ n(u, v) = n^-(u) \cdot n^+(v), \quad (u, v) \in R \]

where

\[ n^-(u) = \begin{cases} 1 & u = s \\ \sum_{v: v \rightarrow u} n^-(v) & \text{otherwise} \end{cases} \]

and

\[ n^+(u) = \begin{cases} 1 & u = t \\ \sum_{v: u \rightarrow v} n^+(v) & \text{otherwise} \end{cases} \]

This is the basis of an efficient algorithm for computing \( n(u, v) \) – after the topological sort of the graph we can compute, using the above relations in topological order, the weights in time of order \( O(m) \), \( m = |R| \). The topological order ensures that all the quantities in the right sides of the above equalities are already computed when needed.
Hummon and Doreian indices and SPC

The Hummon and Doreian indices are defined as follows:

- **search path link count** (SPLC) method: \( w_l(u, v) \) equals the number of “all possible search paths through the network emanating from an origin node” through the arc \((u, v) \in R\).

- **search path node pair** (SPNP) method: \( w_p(u, v) \) “accounts for all connected node pairs along the paths through the arc \((u, v) \in R\)”.

We get the SPLC weights by applying the SPC method on the network obtained from a given standardized network by linking the source \(s\) by an arc to each nonminimal vertex from \(U\); and the SPNP weights by applying the SPC method on the network obtained from the SPLC network by additionally linking by an arc each nonmaximal vertex from \(U\) to the sink \(t\).
The quantities used to compute the arc weights $w$ can be used also to define the corresponding node weights $t$

\[
  t_c(u) = n^- (u) \cdot n^+ (u) \\
  t_i(u) = n_i^- (u) \cdot n_i^+ (u) \\
  t_p(u) = n_p^- (u) \cdot n_p^+ (u)
\]

They are counting the number of paths of selected type through the node $u$. 
Properties of SPC weights

The values of counters $n(u, v)$ form a flow in the citation network – the Kirchoff’s node law holds: For every node $u$ in a standardized citation network incoming flow = outgoing flow:

$$\sum_{v: vRu} n(v, u) = \sum_{v: uRv} n(u, v) = n^-(u) \cdot n^+(u)$$

The weight $n(t, s)$ equals to the total flow through network and provides a natural normalization of weights

$$w(u, v) = \frac{n(u, v)}{n(t, s)} \Rightarrow 0 \leq w(u, v) \leq 1$$

and if $C$ is a minimal arc-cut-set $\sum_{(u,v) \in C} w(u, v) = 1$.

In large networks the values of weights can grow very large. This should be considered in the implementation of the algorithms.
Nonacyclic citation networks

If there is a cycle in a network then there is also an infinite number of trails between some units. There are some standard approaches to overcome the problem: to introduce some 'aging' factor which makes the total weight of all trails converge to some finite value; or to restrict the definition of a weight to some finite subset of trails – for example paths or geodesics. But, new problems arise: What is the right value of the 'aging' factor? Is there an efficient algorithm to count the restricted trails?

The other possibility, since a citation network is usually almost acyclic, is to transform it into an acyclic network

• by identification (shrinking) of cyclic groups (nontrivial strong components), or

• by deleting some arcs, or

• by transformations such as the 'preprint' transformation.
The preprint transformation is based on the following idea: Each paper from a strong component is duplicated with its 'preprint' version. The papers inside strong component cite preprints.

Large strong components in citation network are unlikely – their presence usually indicates an error in the data.

An exception from this rule is the HEP citation network of High Energy Particle Physics literature from arXiv. In it different versions of the same paper are treated as a unit. This leads to large strongly connected components. The idea of preprint transformation could be used also in this case to eliminate cycles.
Another way to measure the importance of nodes and arcs in acyclic networks is the following. Let $\mathcal{N} = (\mathcal{V}, \mathcal{A})$ be a standardized acyclic network with source $s \in \mathcal{V}$ and sink $t \in \mathcal{V}$. The node potential, $p(v)$, is defined by

$$p(s) = 1 \quad \text{and} \quad p(v) = \sum_{u : (u, v) \in \mathcal{A}} \frac{p(u)}{\text{outdeg}(u)}$$

The flow on the arc $(u, v)$ is defined as $\varphi(u, v) = \frac{p(u)}{\text{outdeg}(u)}$. It follows immediately that

$$p(v) = \sum_{u : (u, v) \in \mathcal{A}} \varphi(u, v)$$

and also,

$$\sum_{u : (v, u) \in \mathcal{A}} \varphi(v, u) = \sum_{u : (v, u) \in \mathcal{A}} \frac{p(v)}{\text{outdeg}(v)} = \frac{p(v)}{\text{outdeg}(v)} \sum_{u : (v, u) \in \mathcal{A}} 1 = p(v)$$
Therefore, for each $v \in V$

$$\sum_{u: (u, v) \in A} \varphi(u, v) = \sum_{u: (v, u) \in A} \varphi(v, u) = p(v)$$

which states that Kirchoff’s law holds for the flow $\varphi$.

The probabilistic interpretation of flows has two parts:

1. The node potential of $v$, $p(v)$, is equal to the probability that a random walk starting in the source $s$ goes through the node $v$, and

2. The arc flow on $(u, v)$, $\varphi(u, v)$, is equal to the probability that a random walk starting in the source, $s$, goes through the arc $(u, v)$.

Note that the measures $p$ and $\varphi$ consider only “users” (future) and do not depend on the past.
probabilistic flow

SN5 citation network, flows multiplied with $10^6$

<table>
<thead>
<tr>
<th>SN5</th>
<th>Citation network</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BARABASI_A(1999)286:509</td>
<td>2481.1796</td>
</tr>
<tr>
<td>4</td>
<td>WASSERMA_S(1994)</td>
<td>1807.7400</td>
</tr>
<tr>
<td>5</td>
<td>RONAYNE_J(1987)</td>
<td>1697.2066</td>
</tr>
<tr>
<td>6</td>
<td>WANT_R(1992)10:91</td>
<td>1694.8577</td>
</tr>
<tr>
<td>7</td>
<td>JEONG_H(2001)411:41</td>
<td>1656.5485</td>
</tr>
<tr>
<td>8</td>
<td>FREEMAN_L(1979)1:215</td>
<td>1559.2715</td>
</tr>
<tr>
<td>10</td>
<td>HOLBEN_B(1998)66:1</td>
<td>1494.6278</td>
</tr>
<tr>
<td>13</td>
<td>FREEMAN_L(1977)40:35</td>
<td>1083.6487</td>
</tr>
<tr>
<td>15</td>
<td>[ANONYMO(2011)]</td>
<td>940.7750</td>
</tr>
<tr>
<td>16</td>
<td>SELBY_P(1996)348:313</td>
<td>884.8034</td>
</tr>
<tr>
<td>17</td>
<td>ZADEH_L(1997)90:111</td>
<td>873.1572</td>
</tr>
<tr>
<td>18</td>
<td>[ANONYMO(2009)]</td>
<td>808.5712</td>
</tr>
<tr>
<td>19</td>
<td>[ANONYMO(2010)]</td>
<td>789.0410</td>
</tr>
<tr>
<td>20</td>
<td>STROGATZ_S(2001)410:268</td>
<td>785.9130</td>
</tr>
<tr>
<td>21</td>
<td>BOCCALET_S(2006)424:175</td>
<td>782.6379</td>
</tr>
<tr>
<td>22</td>
<td>GRANOVET(1973)78:1360</td>
<td>777.3402</td>
</tr>
<tr>
<td>23</td>
<td>PARTON_R(1994)127:1199</td>
<td>751.0084</td>
</tr>
<tr>
<td>24</td>
<td>GAREY_M(1979)</td>
<td>734.1673</td>
</tr>
<tr>
<td>28</td>
<td>BOYD_S(2004)</td>
<td>681.9335</td>
</tr>
<tr>
<td>30</td>
<td>AMARAL_L(2000)97:1149</td>
<td>659.7386</td>
</tr>
<tr>
<td>31</td>
<td>CARZANIG_A(2001)19:332</td>
<td>658.6667</td>
</tr>
<tr>
<td>32</td>
<td>BURT_R(1992)</td>
<td>635.2949</td>
</tr>
<tr>
<td>33</td>
<td>[ANONYMO(2008)]</td>
<td>621.7552</td>
</tr>
<tr>
<td>34</td>
<td>JADBABAI_A(2003)48:988</td>
<td>599.2536</td>
</tr>
<tr>
<td>35</td>
<td>BORGATTI_S(2002)</td>
<td>594.1600</td>
</tr>
<tr>
<td>36</td>
<td>ALBERT_R(1999)401:130</td>
<td>589.2179</td>
</tr>
<tr>
<td>37</td>
<td>NEUMAN_M(2001)98:404</td>
<td>584.6247</td>
</tr>
<tr>
<td>38</td>
<td>[ANONYMO(2006)]</td>
<td>570.9648</td>
</tr>
<tr>
<td>40</td>
<td>KATZELA_I(1996)3:10</td>
<td>558.6965</td>
</tr>
<tr>
<td>41</td>
<td>LYNN_D(2008)4</td>
<td>512.6603</td>
</tr>
<tr>
<td>42</td>
<td>BLUMENTH_D(1994)82:1650</td>
<td>509.7068</td>
</tr>
<tr>
<td>43</td>
<td>[ANONYMO(2007)]</td>
<td>508.8429</td>
</tr>
<tr>
<td>44</td>
<td>DOROGOVT_S(2002)51:1079</td>
<td>505.0996</td>
</tr>
<tr>
<td>45</td>
<td>SCOTT_J(2000)</td>
<td>497.5110</td>
</tr>
<tr>
<td>46</td>
<td>RAVASZ_E(2002)297:1551</td>
<td>496.6379</td>
</tr>
<tr>
<td>47</td>
<td>UETZ_P(2000)403:623</td>
<td>496.3690</td>
</tr>
<tr>
<td>48</td>
<td>ERDOS_P(1959)6:290</td>
<td>483.7304</td>
</tr>
<tr>
<td>49</td>
<td>DIAMOND_D(2008)75:606</td>
<td>466.9622</td>
</tr>
<tr>
<td>50</td>
<td>VANLANSC_J(2001)91:1574</td>
<td>465.9244</td>
</tr>
</tbody>
</table>
Another example of acyclic networks are genealogies. In 'Sources' we already described the following network representations of genealogies:

Ore graph, $p$-graph, and bipartite $p$-graph
Properties of representations

$p$-graphs and bipartite $p$-graphs have many advantages:

- there are less nodes and links in $p$-graphs than in corresponding Ore graphs;
- $p$-graphs are directed, acyclic networks;
- every semi-cycle of the $p$-graph corresponds to a relinking marriage. There exist two types of relinking marriages: blood marriage: e.g., marriage among brother and sister, and non-blood marriage: e.g., two brothers marry two sisters from another family.
- $p$-graphs are more suitable for analyses.

Bipartite $p$-graphs have an additional advantage: we can distinguish between a married uncle and a remarriage of a father. This property enables us, for example, to find marriages between half-brothers and half-sisters.
Genealogies are sparse networks

A genealogy is **regular** if every person in it has at most two parents. Genealogies are **sparse** networks – number of links is of the same order as the number of nodes.

For a regular Ore genealogy \((\mathcal{V}, (\mathcal{A}, \mathcal{E}))\) we have:

\[
|\mathcal{A}| \leq 2|\mathcal{V}|, \quad |\mathcal{E}| \leq \frac{1}{2}|\mathcal{V}|, \quad |\mathcal{L}| = |\mathcal{A}| + |\mathcal{E}| \leq \frac{5}{2}|\mathcal{V}|
\]

\(p\)-graphs are almost trees – deviations from trees are caused by relinking marriages \((\mathcal{V}_p, \mathcal{A}_p – \text{nodes and arcs of } p\text{-graph, } n_{mult} – \# \text{ of nodes with multiple marriages})\):

\[
|\mathcal{V}_p| = |\mathcal{V}| - |\mathcal{E}| + n_{mult}, \quad |\mathcal{V}| \geq |\mathcal{V}_p| \geq \frac{1}{2}|\mathcal{V}|, \quad |\mathcal{A}_p| \leq 2|\mathcal{V}_p|
\]

and for a bipartite \(p\)-graph, we have

\[
|\mathcal{V}| \leq |\mathcal{V}_b| \leq \frac{3}{2}|\mathcal{V}|, \quad |\mathcal{A}_b| \leq 2|\mathcal{V}| + n_{mult}
\]
### Number of nodes and links in Ore and $p$-graphs

| data       | $|\mathcal{V}|$ | $|\mathcal{E}|$ | $|\mathcal{A}|$ | $|\mathcal{E}|/|\mathcal{V}|$ | $|\mathcal{V}_i|$ | $n_{\text{mult}}$ | $|\mathcal{V}_p|$ | $|\mathcal{A}_p|$ | $|\mathcal{A}_p|/|\mathcal{V}_p|$ |
|------------|----------------|----------------|----------------|-----------------------------|----------------|----------------|----------------|----------------|---------------------------|
| Drame      | 29606          | 8256           | 41814          | 1.69                       | 13937          | 843            | 22193          | 21862          | 0.99                      |
| Hawlina    | 7405           | 2406           | 9908           | 1.66                       | 2808           | 215            | 5214           | 5306           | 1.02                      |
| Marcus     | 702            | 215            | 919            | 1.62                       | 292            | 20             | 507            | 496            | 0.98                      |
| Mazol      | 2532           | 856            | 3347           | 1.66                       | 894            | 74             | 1750           | 1794           | 1.03                      |
| President  | 2145           | 978            | 2223           | 1.49                       | 282            | 93             | 1260           | 1222           | 0.97                      |
| Royale     | 17774          | 7382           | 25822          | 1.87                       | 4441           | 1431           | 11823          | 15063          | 1.27                      |
| Loka       | 47956          | 14154          | 68052          | 1.71                       | 21074          | 1426           | 35228          | 36192          | 1.03                      |
| Silba      | 6427           | 2217           | 9627           | 1.84                       | 2263           | 270            | 4480           | 5281           | 1.18                      |
| Ragusa     | 5999           | 2002           | 9315           | 1.89                       | 2347           | 379            | 4376           | 5336           | 1.22                      |
| Tur        | 1269           | 407            | 1987           | 1.89                       | 549            | 94             | 956            | 1114           | 1.17                      |
| Royal92    | 3010           | 1138           | 3724           | 1.62                       | 1003           | 269            | 2141           | 2259           | 1.06                      |
| Little     | 25968          | 8778           | 34640          | 1.67                       | 8412           |                |                |                | 1.01                      |
| Mumma      | 34224          | 11334          | 45565          | 1.66                       | 11556          |                |                |                | 1.00                      |
| Tilltson   | 42559          | 12796          | 54043          | 1.57                       | 16967          |                |                |                | 1.00                      |
Relinking index

Let \( n \) denotes number of nodes in \( p \)-graph, \( m \) number of arcs, \( k \) number of weakly connected components, and \( M \) number of maximal nodes (nodes having output degree 0, \( M \geq 1 \)).

The relinking index is defined as:

\[
RI = \frac{k + m - n}{k + n - 2M}
\]

For a trivial graph (having only one node) we define \( RI = 0 \).

It holds \( 0 \leq RI \leq 1 \). \( RI = 0 \) iff network is a forest.
Pattern searching

If a selected pattern determined by a given graph does not occur frequently in a sparse network the straightforward backtracking algorithm applied for pattern searching finds all appearances of the pattern very fast even in the case of very large networks. Pattern searching was successfully applied to searching for patterns of atoms in molecule (carbon rings) and searching for relinking marriages in genealogies.

Three connected relinking marriages in the genealogy (represented as a $p$-graph) of ragusan noble families. A solid arc indicates the _is a son of _ relation, and a dotted arc indicates the _is a daughter of _ relation. In all three patterns a brother and a sister from one family found their partners in the same other family.
To speed up the search or to consider some additional properties of the pattern, a user can set some additional options:

- nodes in network should match with nodes in pattern in some nominal, ordinal or numerical property (for example, type of atom in molecule);

- values of edges must match (for example, edges representing male/female links in the case of \( p \)-graphs);

- the first node in the pattern can be selected only from a given subset of nodes in the network.
Relinking patterns in \( p \)-graphs

All possible relinking marriages in \( p \)-graphs with 2 to 6 nodes. Patterns are labeled as follows:

- second character: number of nodes in pattern (2, 3, 4, 5, or 6).
- last character: identifier (if the two first characters are identical).

Patterns denoted by A are exactly the blood marriages. In every pattern the number of first nodes is equal to the number of last nodes.

frag16.paj
Most of the relinking marriages happened in the genealogy of Turkish nomads; the second is Ragusa while in other genealogies they are much less frequent.
Bipartite $p$-graphs: Marriage among half-cousins
Let $G = (V, R)$ be a simple directed graph without loops. A triad is a subgraph induced by a given set of three nodes. There are 16 nonisomorphic (types of) triads. They can be partitioned into three basic types:

- the null triad 003;
- dyadic triads 012 and 102; and
- connected triads: 111D, 201, 210, 300, 021D, 111U, 120D, 021U, 030T, 120U, 021C, 030C and 120C.
Several properties of a graph can be expressed in terms of its **triadic spectrum** – distribution of all its triads. It also provides ingredients for $p^*$ network models.

A direct approach to determine the triadic spectrum is of order $O(n^3)$; but in most large graphs it can be determined much faster.