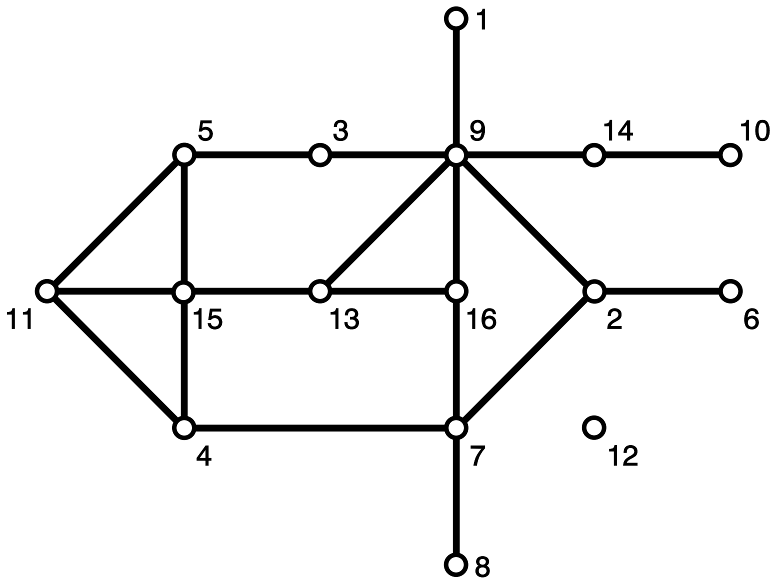


ISEG
Technical University of Lisbon

Introductory Workshop to Network Analysis of Texts

Properties and weights

Vladimir Batagelj

University of Ljubljana

Lisbon, Portugal: 2nd to 5th February 2004

organized by **SOCIUS** – Research Centre on Economic Sociology and the Sociology of Organisations

Outline

Important vertices in network

It seems that the most important distinction between different vertex *indices* is based on the view/decision whether the network is considered directed or undirected. This gives us two main types of indices:

- directed case: measures of *importance*; with two subgroups: measures of *influence*, based on out-going arcs; and measures of *support*, based on incoming arcs;
- undirected case: measures of *centrality*, based on all links.

For undirected networks all three types of measures coincide.

If we change the direction of all arcs (replace the relation with its inverse relation) the measure of influence becomes a measure of support, and vice versa.

... Important vertices in network

The real meaning of measure of importance depends on the relation described by a network. For example the most 'important' person for the relation '__ doesn't like to work with __' is in fact the least popular person.

Removal of an important vertex from a network produces a substantial change in structure/functioning of the network.

Normalization

Let $p : V \to \mathbb{R}$ be an index in network $\mathbf{N} = (V, L)$. If we want to compare indices p over different networks we have to make them comparable. Usually we try to achieve this by *normalization* of p.

Let $\mathbf{N} \in \mathcal{N}(V)$, where $\mathcal{N}(V)$ is a selected set of networks over the same set of vertices V,

$$p_{max} = \max_{\mathbf{N}\in\mathcal{N}(V)} \max_{v\in V} p_{\mathbf{N}}(v) \qquad \text{and} \qquad p_{min} = \min_{\mathbf{N}\in\mathcal{N}(V)} \min_{v\in V} p_{\mathbf{N}}(v)$$

then we define the normalized index as

$$p'(v) = \frac{p(v) - p_{min}}{p_{max} - p_{min}} \in [0, 1]$$

Degrees

The simplest index are the degrees of vertices. Since for simple networks $\deg_{min} = 0$ and $\deg_{max} = n - 1$, the corresponding normalized indices are

centrality $$\deg'(v) = \frac{\deg(v)}{n-1}$$

and similary

support $$\mathrm{indeg}'(v) = \frac{\mathrm{indeg}(v)}{n}$$

influence $$\mathrm{outdeg}'(v) = \frac{\mathrm{outdeg}(v)}{n}$$

Instead of degrees in original network we can consider also the degrees with respect to the reachability relation (transitive closure).

Closeness

Most indices are based on the distance $d(u,v)$ between vertices in a network $\mathbf{N} = (V, L)$. Two such indices are

radius $r(v) = \max_{u \in V} d(v,u)$

total closeness $S(v) = \sum_{u \in V} d(v,u)$

These two indices are measures of influence – to get measures of support we have to replace in definitions $d(u,v)$ with $d(v,u)$.

If the network is not strongly connected r_{max} and S_{max} equal ∞. Sabidussi (1966) introduced a related measure $1/S(v)$; or in its normalized form

closeness $$cl(v) = \frac{n-1}{\sum_{u \in V} d(v,u)}$$

$D = \max_{u,v \in V} d(v,u)$ is called the *diameter* of network.

Betweeness

Important are also the vertices that can control the information flow in the network. If we assume that this flow uses only the shortest paths (geodesics) we get a measure of *betweeness* (Anthonisse 1971, Freeman 1977)

$$b(v) = \frac{1}{(n-1)(n-2)} \sum_{\substack{u,t \in V: g_{u,t} > 0 \\ u \neq v, t \neq v, u \neq t}} \frac{g_{u,t}(v)}{g_{u,t}}$$

where $g_{u,t}$ is the number of geodesics from u to t; and $g_{u,t}(v)$ is the number of those among them that pass through vertex v.

If we know matrices $[d_{u,v}]$ and $[g_{u,v}]$ we can determine also $g_{u,v}(t)$ by:

$$g_{u,v}(t) = \begin{cases} g_{u,t} \cdot g_{t,v} & d_{u,t} + d_{t,v} = d_{u,v} \\ 0 & \text{otherwise} \end{cases}$$

Hubs and authorities

To each vertex v of a network $\mathbf{N} = (V, L)$ we assign two values: quality of its content (*authority*) x_v and quality of its references (*hub*) y_v.

A good authority is selected by good hubs; and good hub points to good authorities.

$$x_v = \sum_{u:(u,v)\in L} y_u \qquad \text{and} \qquad y_v = \sum_{u:(v,u)\in L} x_u$$

Let $\mathbf{W}$ be a matrix of network $\mathbf{N}$ and $\mathbf{x}$ and $\mathbf{y}$ authority and hub vectors. Then we can write these two relations as $\mathbf{x} = \mathbf{W}^T\mathbf{y}$ and $\mathbf{y} = \mathbf{W}\mathbf{x}$.

We start with $\mathbf{y} = [1, 1, \ldots, 1]$ and then compute new vectors $\mathbf{x}$ and $\mathbf{y}$. After each step we normalize both vectors. We repeat this until they stabilize.

We can show that this procedure converges. The limit vector $\mathbf{x}^*$ is the principal eigen vector of matrix $\mathbf{W}^T\mathbf{W}$; and $\mathbf{y}^*$ of matrix $\mathbf{W}\mathbf{W}^T$.

. . . Hubs and authorities

Similar procedures are used in search engines on the web to evaluate the importance of web pages.

PageRank, PageRank / Google, HITS / AltaVista, SALSA, teorija.

Examples: Krebs, Krempl.

Clustering

Clustering in vertex v is usually measured as a quotient between the number of links in subgraph $G^1(v)$ induced by the neighbors of vertex v and the number of links in the complete graph on these vertices:

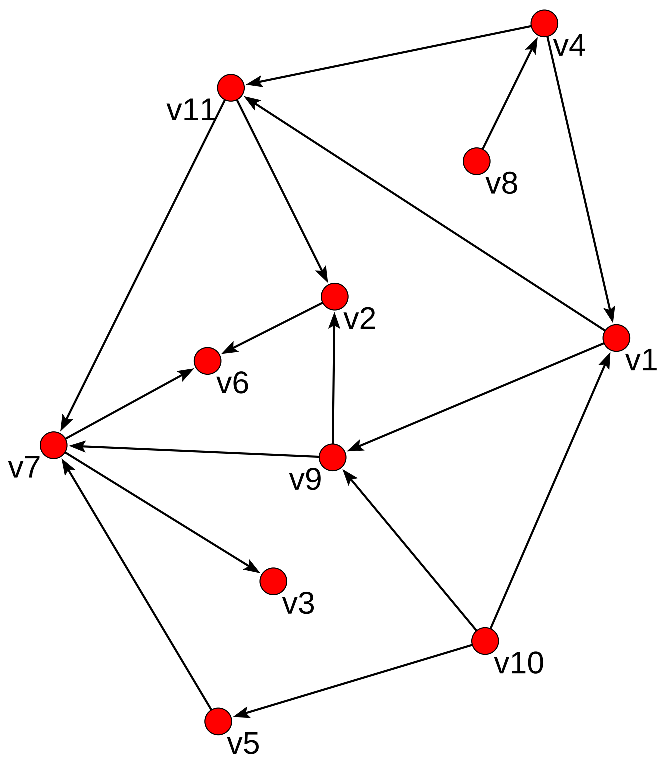
$$C(v) = \begin{cases} \dfrac{2|L(G^1(v))|}{\deg(v)(\deg(v)-1)} & \deg(v) > 1 \\ 0 & \text{otherwise} \end{cases}$$

We can consider also the size of vertex neighborhood by the following correction

$$C_1(v) = \frac{\deg(v)}{\Delta} C(v)$$

where Δ is the maximum degree in graph G. This measure attains its largest value in vertices that belong to an isolated clique of size Δ.

Acyclic networks



acyclic.paj

Network $\mathbf{G} = (V, R)$, $R \subseteq V \times V$ 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 vertex property $p : V \to \mathbb{R}$ (most often time), that is *compatible* with arcs

$$(u, v) \in R \Rightarrow p(u) < p(v)$$

Basic properties of acyclic networks

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

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

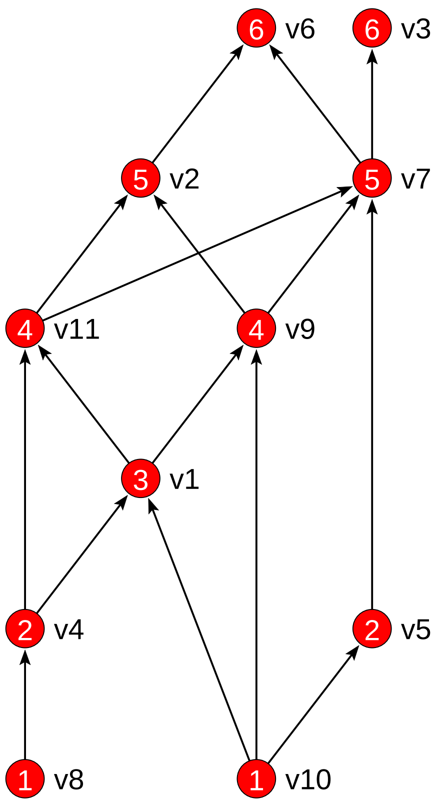
The set of *sources* $\mathrm{Min}_R(V) = \{v : \neg\exists u \in V : (u, v) \in R\}$ and the set of *sinks* $\mathrm{Max}_R(V) = \{v : \neg\exists u \in V : (v, u) \in R\}$ are nonempty (in finite networks).

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 $Q = R \setminus R * \overline{R}$.

Depth

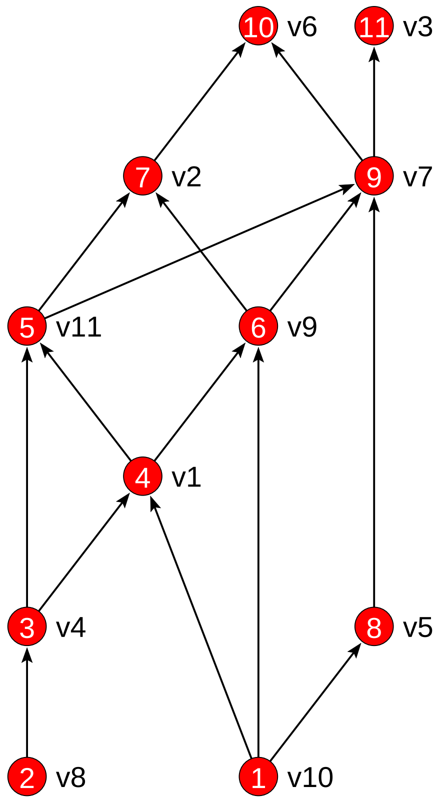


Mapping $h : V \rightarrow \mathbb{N}^+$ is called *depth* or *level* if all differences on the longest path and the initial value equal to 1.

$U \leftarrow V; k \leftarrow 0$
while $U \neq \emptyset$ **do**
  $T \leftarrow \mathrm{Min}_R(U); k \leftarrow k + 1$
  for $v \in T$ **do** $h(v) \leftarrow k$
  $U \leftarrow U \setminus T$

Drawing on levels. Macro `Layers`.

Compatible numberings



Injective mapping $h : V \to 1..|V|$ compatible with relation R is called a *compatible numbering*.

'Topological sort'

$U \leftarrow V$; $k \leftarrow 0$

while $U \neq \emptyset$ **do**

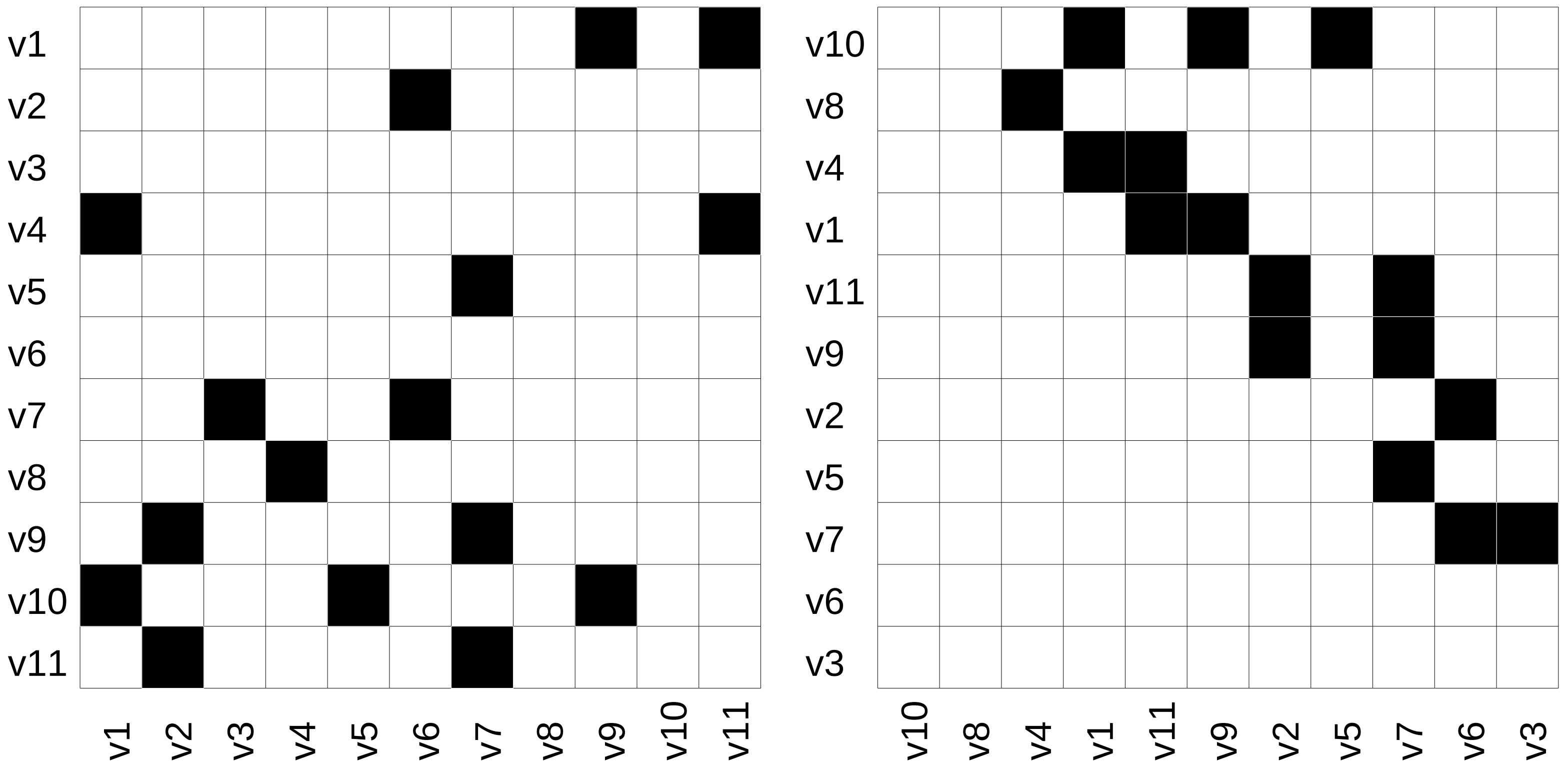
  select $v \in \mathrm{Min}_R(U)$; $k \leftarrow k+1$

  $h(v) \leftarrow k$

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

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

. . . Compatible numberings



Compatible numberings and functions on acyclic networks

Let the function $f : V \rightarrow \mathbb{R}$ be defined in the following way:

- $f(v)$ is knownn in sources $v \in \mathrm{Min}_R(V)$
- $f(v) = F(\{f(u) : uRv\})$

If we compute the values of function f in a sequence determined by a comptible numbering we can compute them in one pass since for each vertex $v \in V$ the values of f needed for its computation are already known.

Compatible numberings – CPM

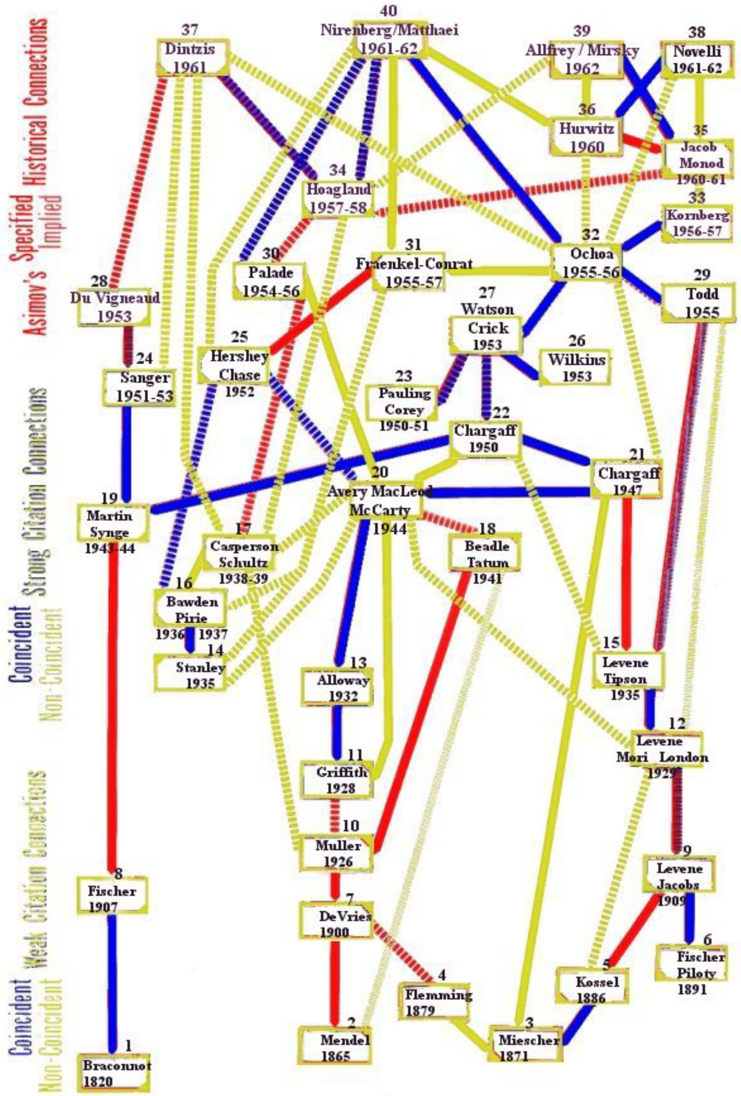
CPM (Critical Path Method): A project consists of tasks. Vertices of a project network represent states of the project and arcs represent tasks. Every project network is acyclic. For each task (u, v) its execution time $t(u, v)$ is known. A task can start only when all the preceeding tasks are finished. 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.

$$T(v) = 0, \qquad v \in \mathrm{Min}_R(V)$$

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

Citation networks



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

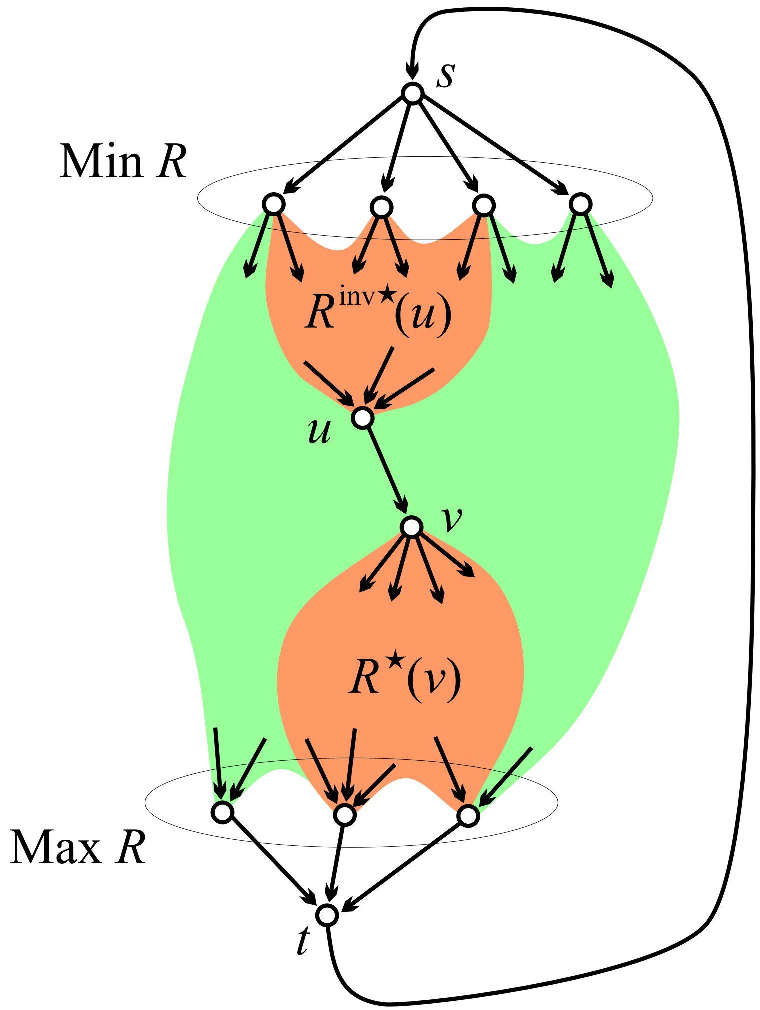
In a given set of units/vertices $\mathbf{U}$ (articles, books, works, etc.) we introduce a *citing relation*/set of arcs $R \subseteq \mathbf{U} \times \mathbf{U}$

$$uRv \equiv v \text{ cites } u$$

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

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 vertices) 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* vertex $s \notin \mathbf{U}$ and a common *sink* vertex $t \notin \mathbf{U}$. 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).

Search path count method



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

$$n(u,v) = n^-(u) \cdot n^+(v), \qquad (u,v) \in R$$

where

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

and

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

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 vertex 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 $\mathbf{U}$; 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 $\mathbf{U}$ to the sink t.

Vertex weights

The quantities used to compute the arc weights w can be used also to define the corresponding vertex weights t

$$
\begin{aligned}
t_c(u) &= n^-(u) \cdot n^+(u) \\
t_l(u) &= n_l^-(u) \cdot n_l^+(u) \\
t_p(u) &= n_p^-(u) \cdot n_p^+(u)
\end{aligned}
$$

They are counting the number of paths of selected type through the vertex u.

Properties of SPC weights

The values of counters $n(u,v)$ form a flow in the citation network – the *Kirchoff's vertex law* holds: For every vertex 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

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

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.

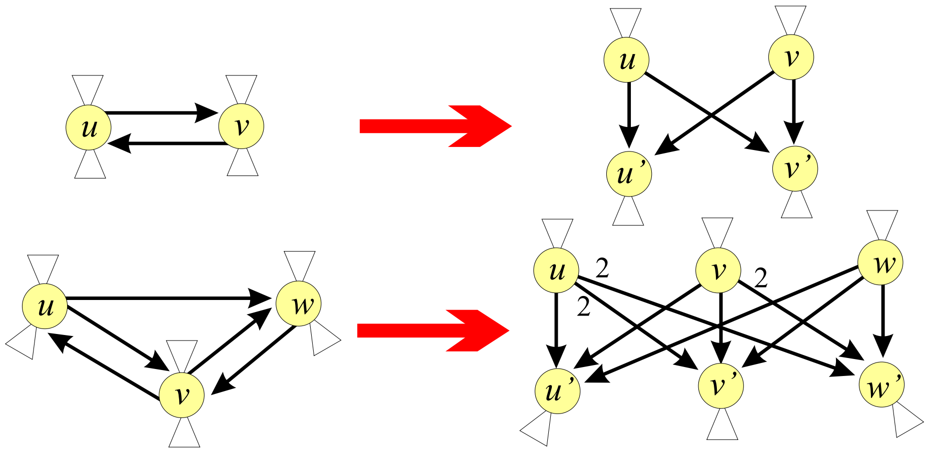
Nonacyclic citation networks

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.

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 can be used also in this case to eliminate cycles.

SOM main subnetwork

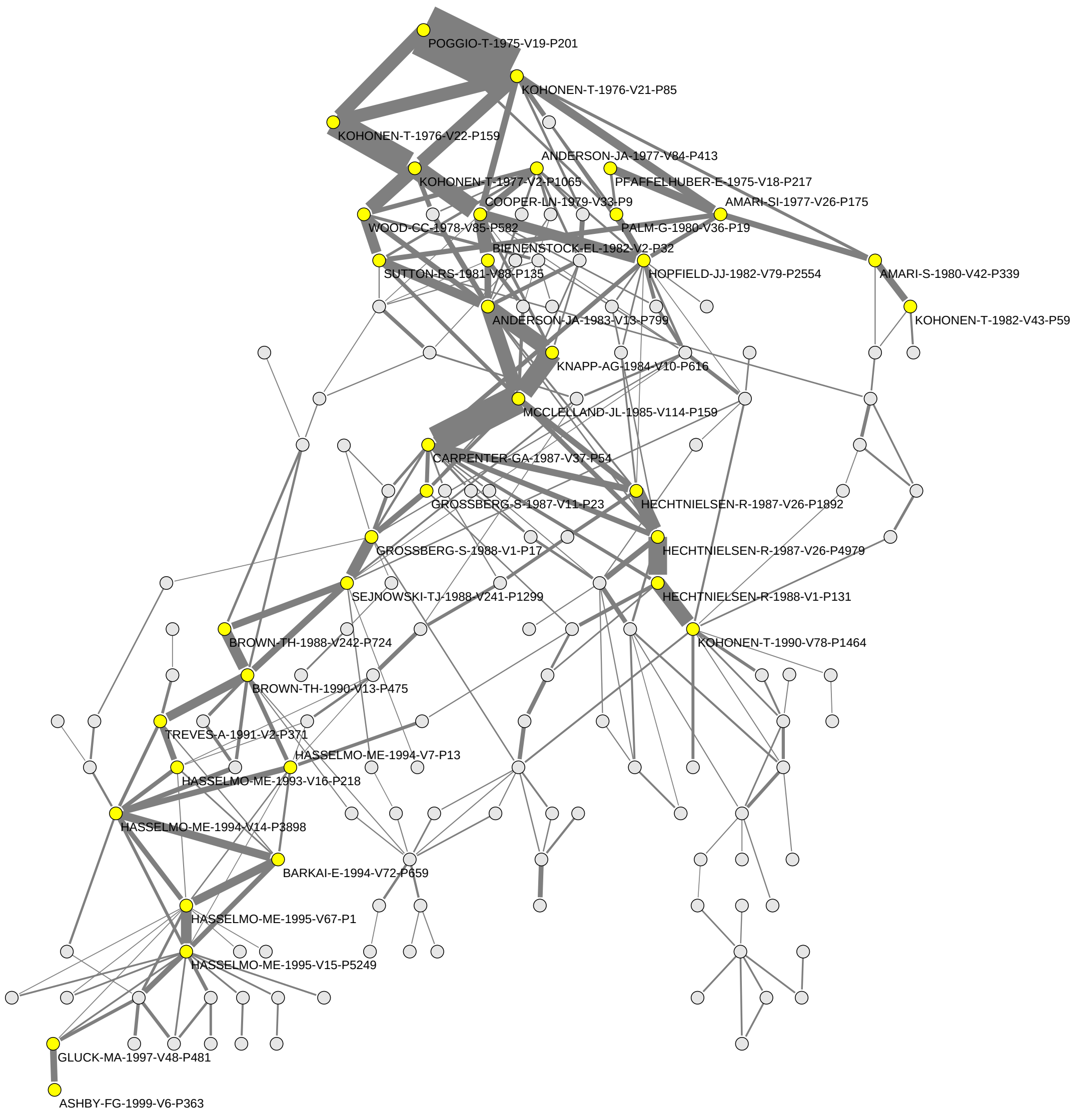
Consider citation network ($n = 4470$, $m = 12731$) on SOM (*self-organizing maps*) literature.

Inspecting the distribution of values of weights on arcs (lines) we select a threshold 0.007 and delete all arcs with weights lower than selected threshold. We delete also all isolated vertices (degree $= 0$) and small ($k = 5$) components. A single component remains. We draw it. We improve the obtained layout manually.

We label only the 'important' vertices – endpoints of arcs with weight at least 0.05.

From the picture we see that there isn't a single stream in the development of SOM field.

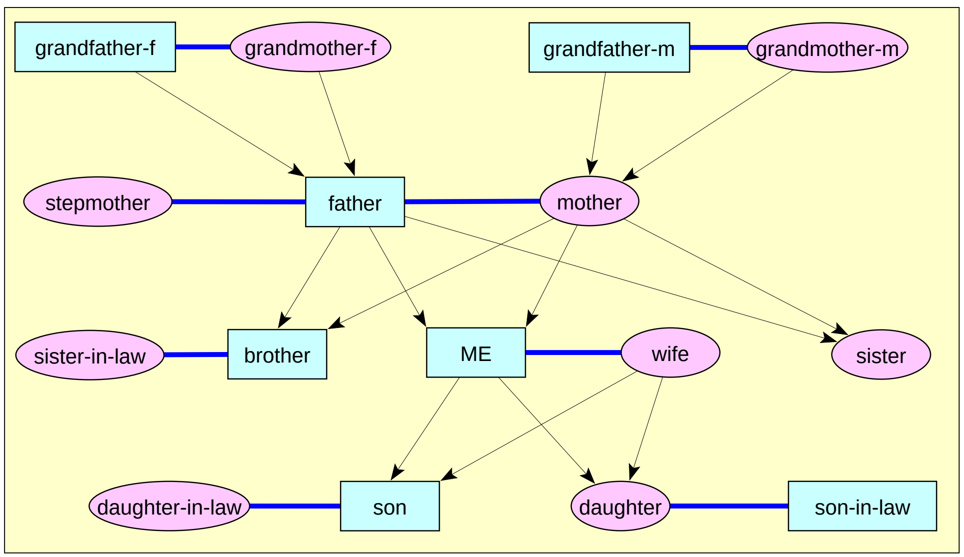
SOM arc-cut subnetwork at level 0.007



Genealogies

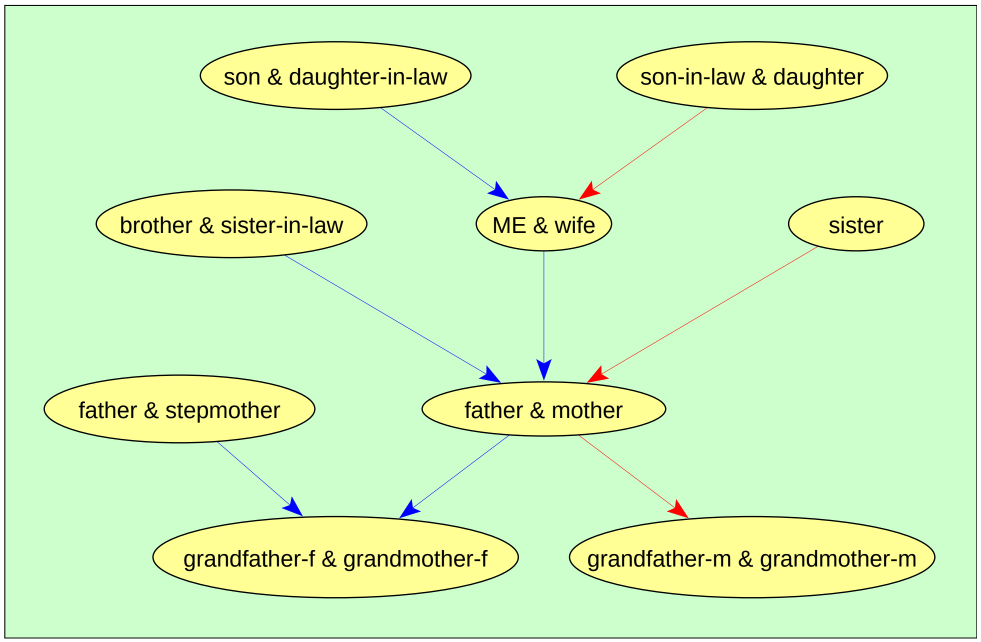
Genealogies are usually available in GEDCOM format. They can be represented as networks in many different ways: as Ore-graph, as p-graph, and as bipartite p-graph.

Ore graph



In Ore graph every person is represented by a vertex, marriages are represented with edges and relation *is a parent of* as arcs pointing from each of the parents to their children.

p-graph



In p-graph vertices represent individuals or couples. In the case that person is not married yet (s)he is represented by vertex, otherwise person is represented with the partner in a common vertex. There are only arcs in p-graphs – they point from children to their parents. p-graphs are acyclic!!!

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 appearences of the pattern very fast even in the case of very large networks.

To speed up the search or to consider some additional properties of the pattern, a user can set some additional options:

- vertices in network should match with vertices in pattern in some nominal, ordinal or numerical property (for example, type of atom in molecula);
- values of edges must match (for example, edges representing male/female links in the case of *p-graphs*);
- the first vertex in the pattern can be selected only from a given subset of vertices in the network.

Marriages among relatives in Ragusa

Pattern searching was successfully applied to searching for patterns of atoms in molecula (carbon rings) and searching for relinking marriages in genealogies.

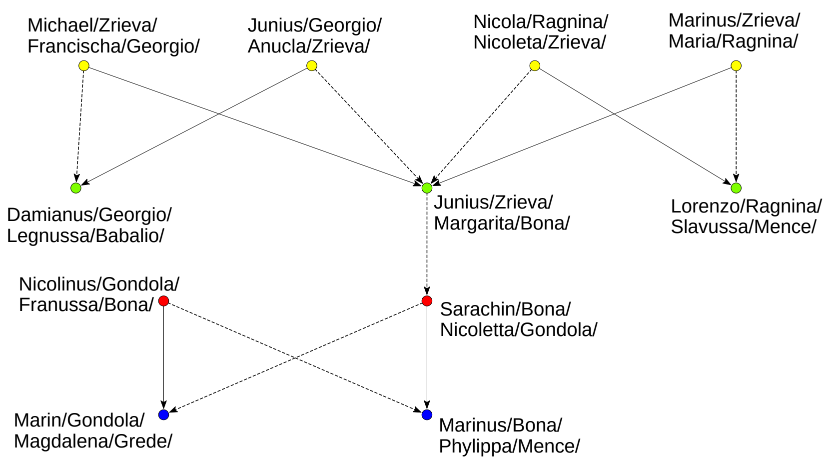
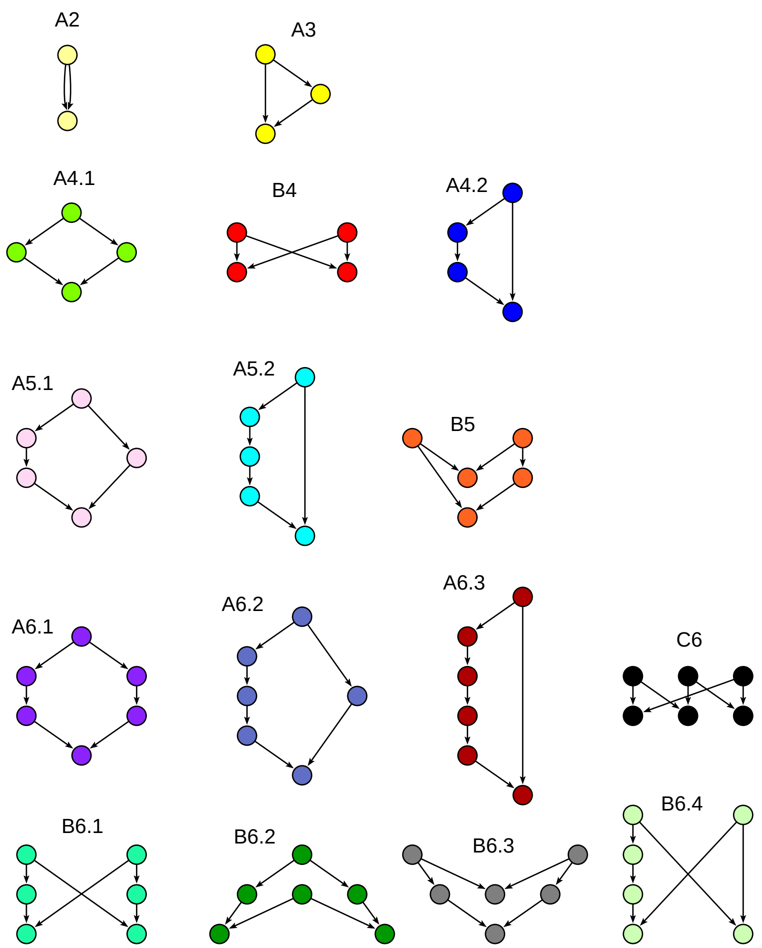


Figure presents three connected relinking marriages which are non-blood marriages found in the genealogy of ragusan noble families. The genealogy is represented as a p-graph. A solid arc indicates the _ *is a son of* _ relation, and a dotted arc indicates the _ *is a daughter of* _ relation. In all three patterns a brother and a sister from one family found their partners in the same other family.

Relinking marriages (p-graphs with 2 to 6 vertices).



In the figure all possible relinking marriages in p-graphs containing from 2 up to 6 vertices are presented. Patterns are marked in the following way:

- first character: A – pattern with one first vertex (vertex without incoming arcs), B – pattern with two first vertices, and C – pattern with three first vertices.
- second character: number of vertices in pattern (2, 3, 4, 5 or 6).
- last character: identifier (if two first characters are equal).

Comparing genealogies

Using frequency distributions for different patterns we can compare different genealogies As example we took four genealogies: `Loka.ged` (genealogy in Škofja Loka district, western part of Slovenia), `Silba.ged` (genealogy of island Silba, Croatia), `Ragusa.ged` (genealogy of Ragusan noble families between 12 and 16 century, Dremelj et al., 2002) and `Tur.ged` (genealogy of Turkish nomads, White et al., 1999).

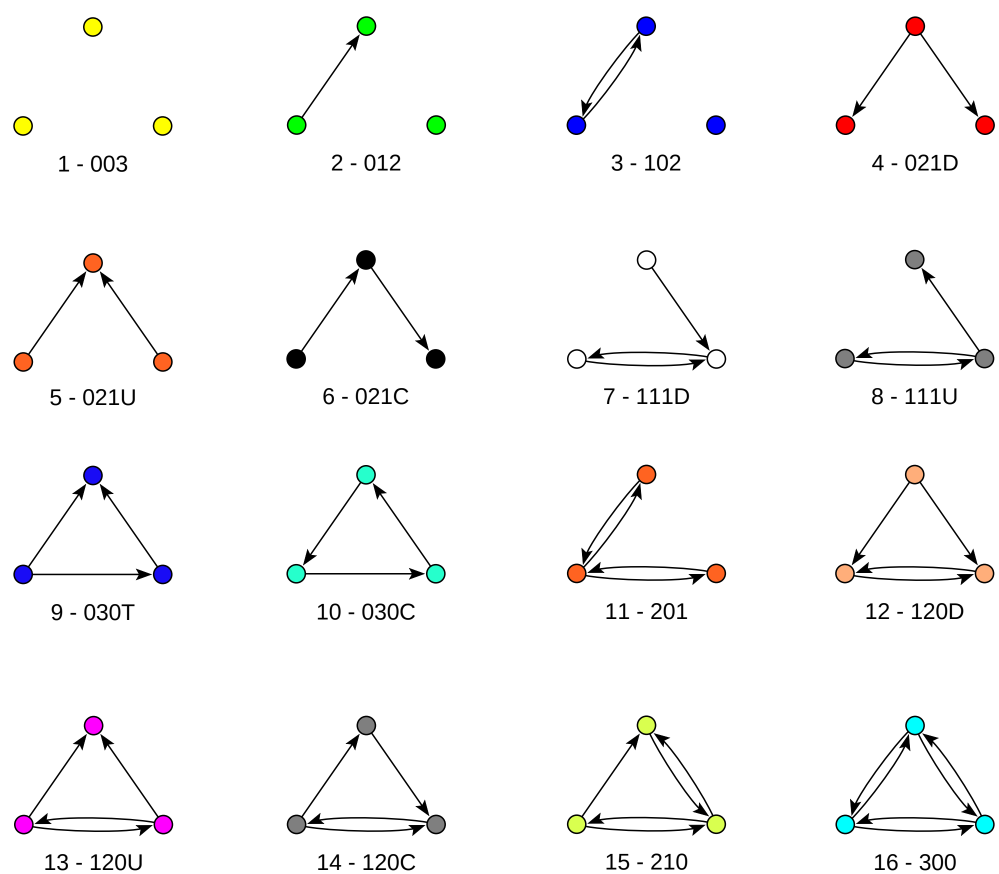
. . . Comparing genealogies

vzorec	Loka	Silba	Ragusa	Tur
A2	1	0	0	0
A3	1	0	0	0
A4.1	11	5	3	65
B4	53	25	21	40
A4.2	0	0	0	0
A5.1	9	7	4	15
A5.2	0	0	0	0
B5	19	11	47	19
A6.1	28	28	2	41
A6.2	0	2	0	0
A6.3	0	0	0	0
C6	10	12	19	15
B6.1	0	1	2	0
B6.2	27	39	63	24
B6.3	47	0	82	25
B6.4	0	0	5	3
No. indi.	47956	6427	5999	1269
p-graph	35228	4480	4376	956
No. bicom.	29	4	2	3
Largest bicom.	4095	1340	1446	250
RI	0.55	0.78	0.74	0.75

From frequency distributions we see:

- Probability of generation jump for more than one generation is very unlikely (patterns A4.2, A5.2 and A6.3 do not appear in any genealogy, pattern A6.2 appears twice in Silba genealogy, pattern B6.4 appears five times in Ragusa and thee times in Tur).
- In Tur there are a lot of marriages of types A4.1 and A6.1.
- For all genealogies number of relinking 'non-blood' marriages (e.g. patterns B4, B5, C6, B6.1, B6.2, B6.3 and B6.4) is much higher than number of blood marriages. There were economic reasons for non-blood relinking marriages: to keep the wealth and power within selected families.

Triads

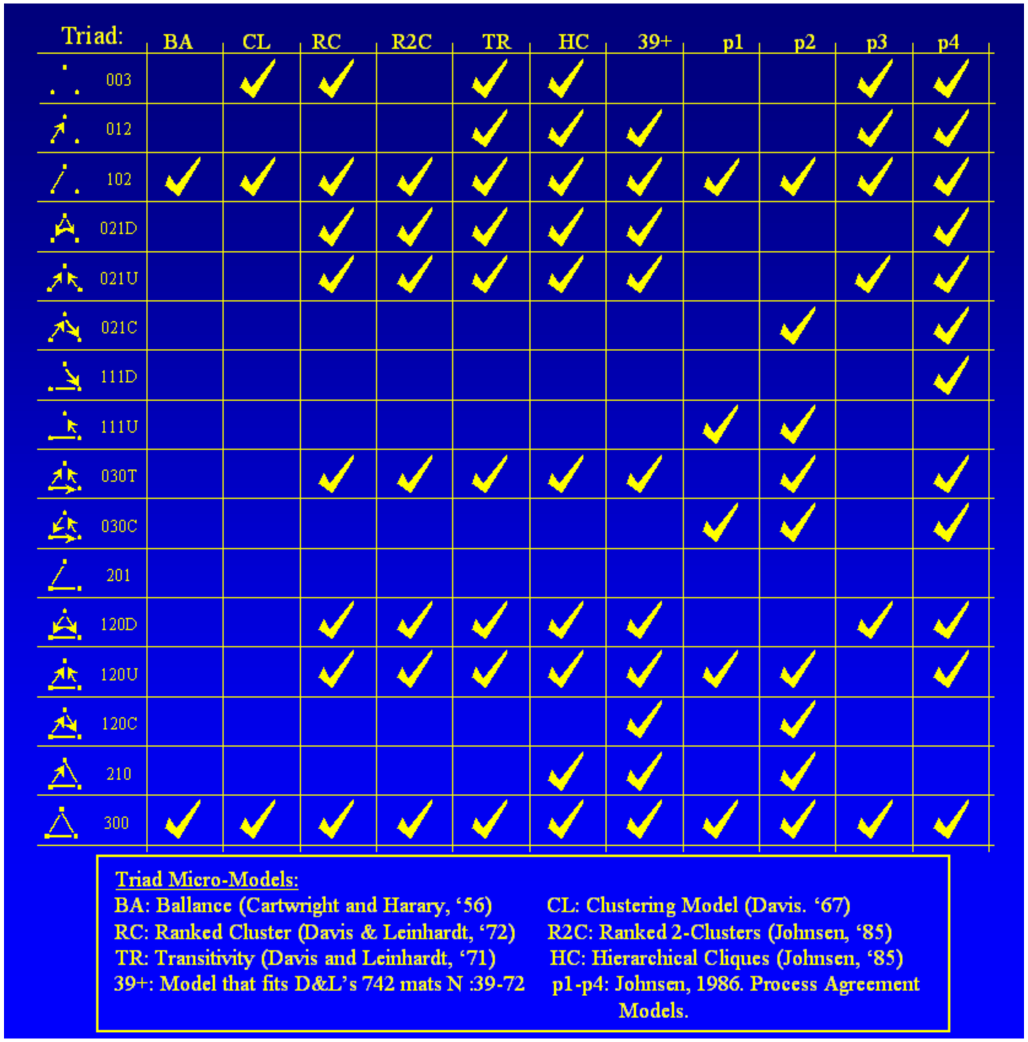


Let $\mathbf{G} = (V, R)$ be a simple directed graph without loops. A *triad* is a subgraph induced by a given set of three vertices.

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.

Triadic spectrum



Triad:	BA	CL	RC	R2C	TR	HC	39+	p1	p2	p3	p4
003		✓	✓		✓	✓				✓	✓
012					✓	✓	✓			✓	✓
102	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
021D			✓	✓	✓	✓	✓				✓
021U			✓	✓	✓	✓	✓			✓	✓
021C									✓		✓
111D											✓
111U								✓	✓		
030T			✓	✓	✓	✓	✓		✓		✓
030C								✓	✓		✓
201											
120D			✓	✓	✓	✓	✓			✓	✓
120U			✓	✓	✓	✓	✓	✓	✓		✓
120C							✓		✓		
210						✓	✓		✓		
300	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Triad Micro-Models:
BA: Ballance (Cartwright and Harary, '56)
CL: Clustering Model (Davis. '67)
RC: Ranked Cluster (Davis & Leinhardt, '72)
R2C: Ranked 2-Clusters (Johnsen, '85)
TR: Transitivity (Davis and Leinhardt, '71)
HC: Hierarchical Cliques (Johnsen, '85)
39+: Model that fits D&L's 742 mats N :39-72
p1-p4: Johnsen, 1986. Process Agreement Models.

Moody

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.

Network based data-mining and normalizations

A *2-mode network* or *affiliation network* is a structure $\mathbf{N} = (U, V, A, w)$, where U and V are disjoint sets of vertices, A is the set of arcs with the initial vertex in the set U and the terminal vertex in the set V, and $w : A \to \mathbb{R}$ is a weight. If no weight is defined we can assume a constant weight $w(u, v) = 1$ for all arcs $(u, v) \in A$. The set A can be viewed also as a relation $A \subseteq U \times V$.

A 2-mode network can be formally represented by rectangular matrix $\mathbf{A} = [a_{uv}]_{U \times V}$.

$$a_{uv} = \begin{cases} w(u,v) & (u,v) \in A \\ 0 & \text{otherwise} \end{cases}$$

Approaches to 2-mode network analysis

For direct analysis of 2-mode networks we can use eigen-vector approach, clustering and blockmodeling.

But most often we transform a 2-mode network into an ordinary (1-mode) network $\mathbf{N}_1 = (U, E_1, w_1)$ or/and $\mathbf{N}_2 = (V, E_2, w_2)$, where E_1 and w_1 are determined by the matrix $\mathbf{A}^{(1)} = \mathbf{A}\mathbf{A}^T$, $a_{uv}^{(1)} = \sum_{z\in V} a_{uz} \cdot a_{zv}^T$. Evidently $a_{uv}^{(1)} = a_{vu}^{(1)}$. There is an edge $\{u, v\} \in E_1$ in $\mathbf{N}_1$ iff $N(u) \cap N(v) \neq \emptyset$. Its weight is $w_1(u, v) = a_{uv}^{(1)}$.

The network $\mathbf{N}_2$ is determined in a similar way by the matrix $\mathbf{A}^{(2)} = \mathbf{A}^T\mathbf{A}$.

The networks $\mathbf{N}_1$ and $\mathbf{N}_2$ are analyzed using standard methods.

Normalizations

The *normalization* approach was developed for quick inspection of (1-mode) networks obtained from 2-mode networks – a kind of network based data-mining.

In networks obtained from large 2-mode networks there are often huge differences in weights. Therefore it is not possible to compare the vertices according to the raw data. First we have to normalize the network to make the weights comparable.

There exist several ways how to do this. Some of them are presented in the following table. They can be used also on other networks.

In the case of networks without loops we define the diagonal weights for undirected networks as the sum of out-diagonal elements in the row (or column) $w_{vv} = \sum_u w_{vu}$ and for directed networks as some mean value of the row and column sum, for example $w_{vv} = \frac{1}{2}(\sum_u w_{vu} + \sum_u w_{uv})$. Usually we assume that the network does not contain any isolated vertex.

. . . Normalizations

$$\mathrm{Geo}_{uv} = \frac{w_{uv}}{\sqrt{w_{uu} w_{vv}}} \qquad \mathrm{GeoDeg}_{uv} = \frac{w_{uv}}{\sqrt{\deg_u \deg_v}}$$

$$\mathrm{Input}_{uv} = \frac{w_{uv}}{w_{vv}} \qquad \mathrm{Output}_{uv} = \frac{w_{uv}}{w_{uu}}$$

$$\mathrm{Min}_{uv} = \frac{w_{uv}}{\min(w_{uu}, w_{vv})} \qquad \mathrm{Max}_{uv} = \frac{w_{uv}}{\max(w_{uu}, w_{vv})}$$

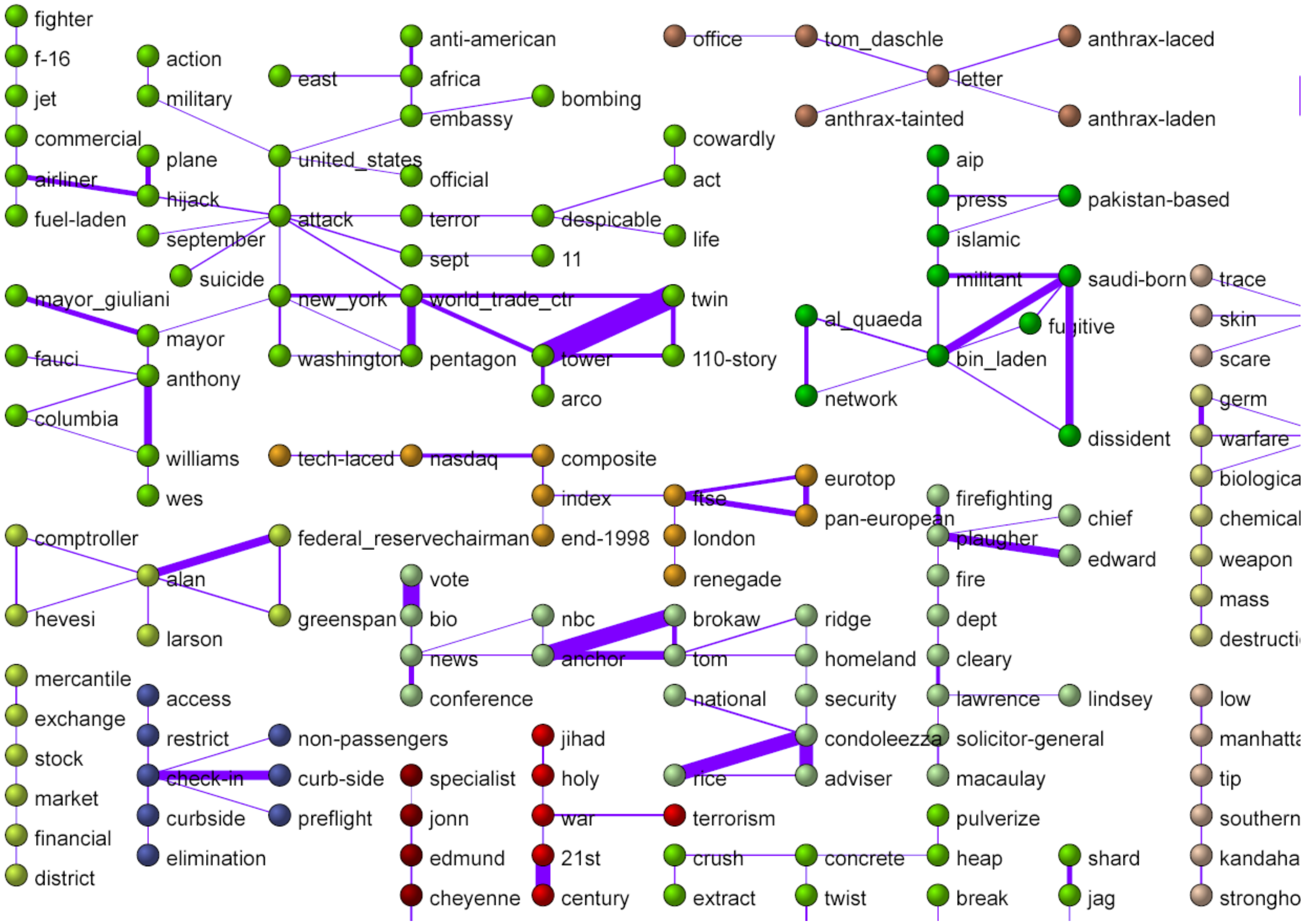
$$\mathrm{MinDir}_{uv} = \begin{cases} \frac{w_{uv}}{w_{uu}} & w_{uu} \leq w_{vv} \\ 0 & \text{otherwise} \end{cases} \qquad \mathrm{MaxDir}_{uv} = \begin{cases} \frac{w_{uv}}{w_{vv}} & w_{uu} \leq w_{vv} \\ 0 & \text{otherwise} \end{cases}$$

After a selected normalization the important parts of network are obtained by line-cutting the normalized network at selected level t and preserving components with at least k vertices.

Slovenian journals and magazins.

Reuters Terror News: GeoDeg, MaxDir, MinDir.

GeoDeg normalization of Reuters terror news network



GeoDeg normalization of Reuters terror news network