

Structure of Networks II

Vladimir Batagelj

University of Ljubljana

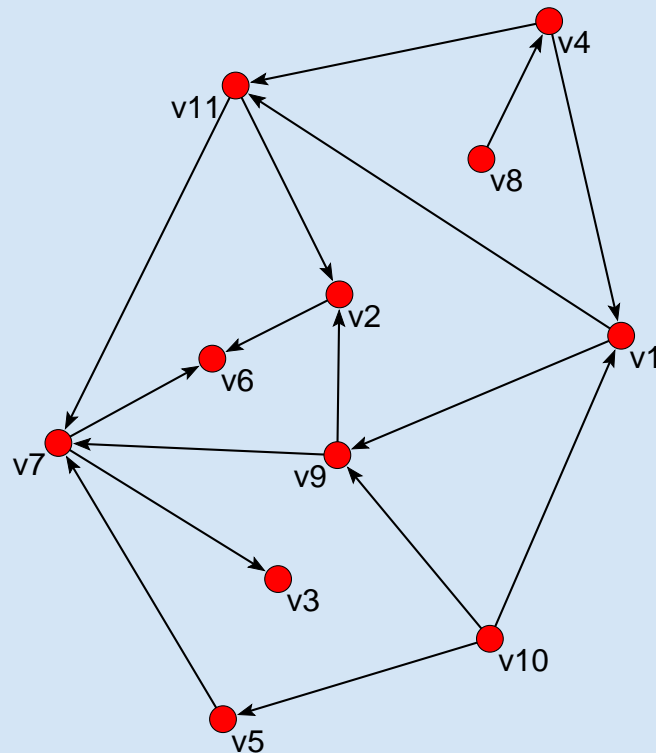
Networks Workshop

NICTA, Sydney, June 2005

Outline

1	Acyclic networks	1
8	Citation networks	8
17	Genealogies	17
22	Pattern searching	22
27	Triads	27
29	Network based data-mining and normalizations	29

Acyclic networks



acyclic.paj

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 vertex property $p : \mathcal{V} \rightarrow \mathbb{R}$ (most often time), that is *compatible* with arcs

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

Basic properties of acyclic networks

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

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

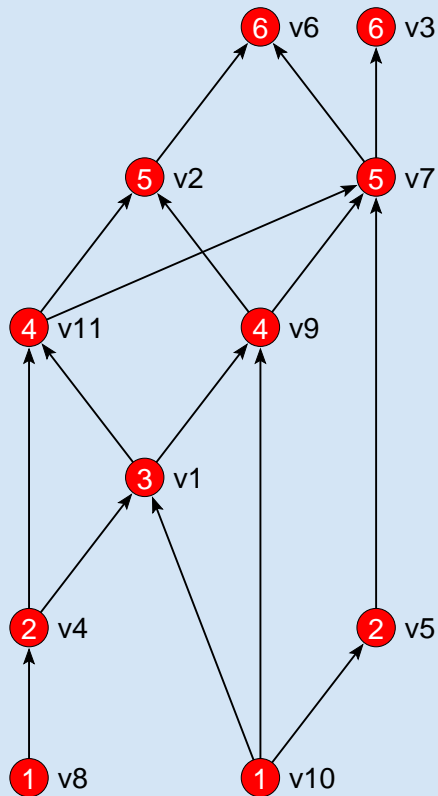
The set of *sources* $\text{Min}_R(\mathcal{V}) = \{v : \neg \exists u \in \mathcal{V} : (u, v) \in R\}$ and the set of *sinks* $\text{Max}_R(\mathcal{V}) = \{v : \neg \exists u \in \mathcal{V} : (v, u) \in R\}$ are nonempty (in finite networks).

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

Depth



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**

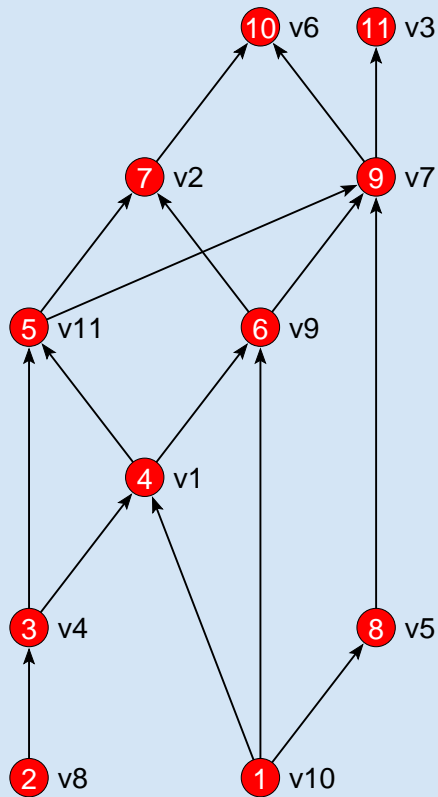
$\mathcal{T} \leftarrow \text{Min}_R(\mathcal{U}); k \leftarrow k + 1$

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

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

Drawing on levels. Macro Layers.

Compatible numberings



Injective mapping $h : \mathcal{V} \rightarrow 1..|\mathcal{V}|$ compatible with relation R is called a *compatible numbering*.

'Topological sort'

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

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

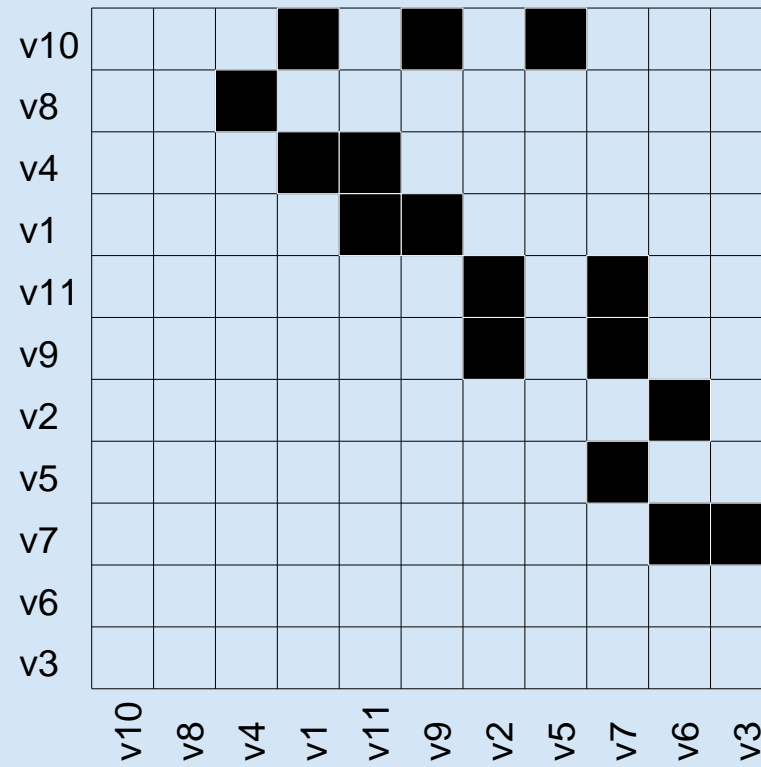
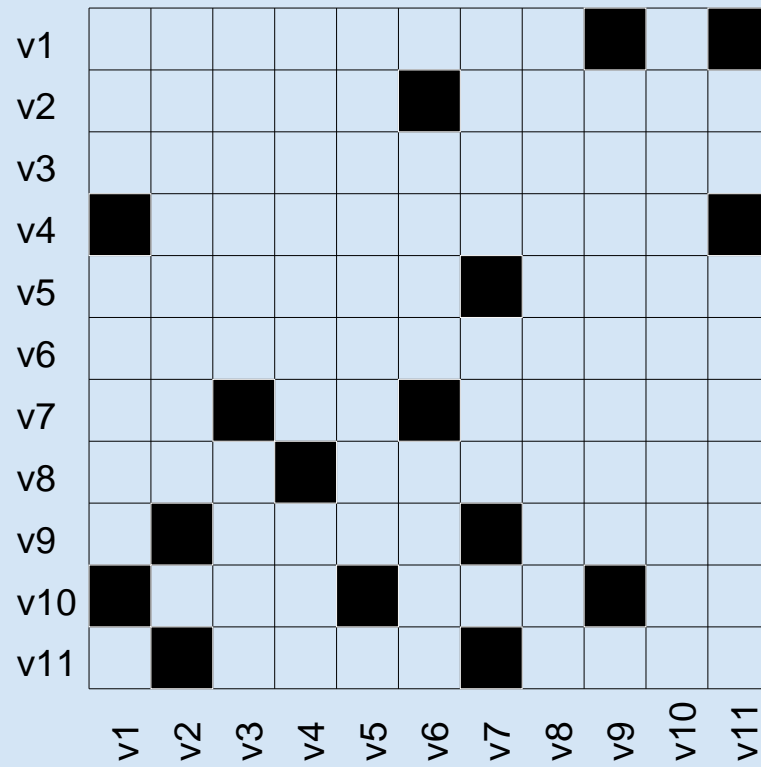
select $v \in \text{Min}_R(\mathcal{U}); k \leftarrow k + 1$

$h(v) \leftarrow k$

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

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

... Compatible numberings



Compatible numberings and functions on acyclic networks

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

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

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

Compatible numberings – CPM

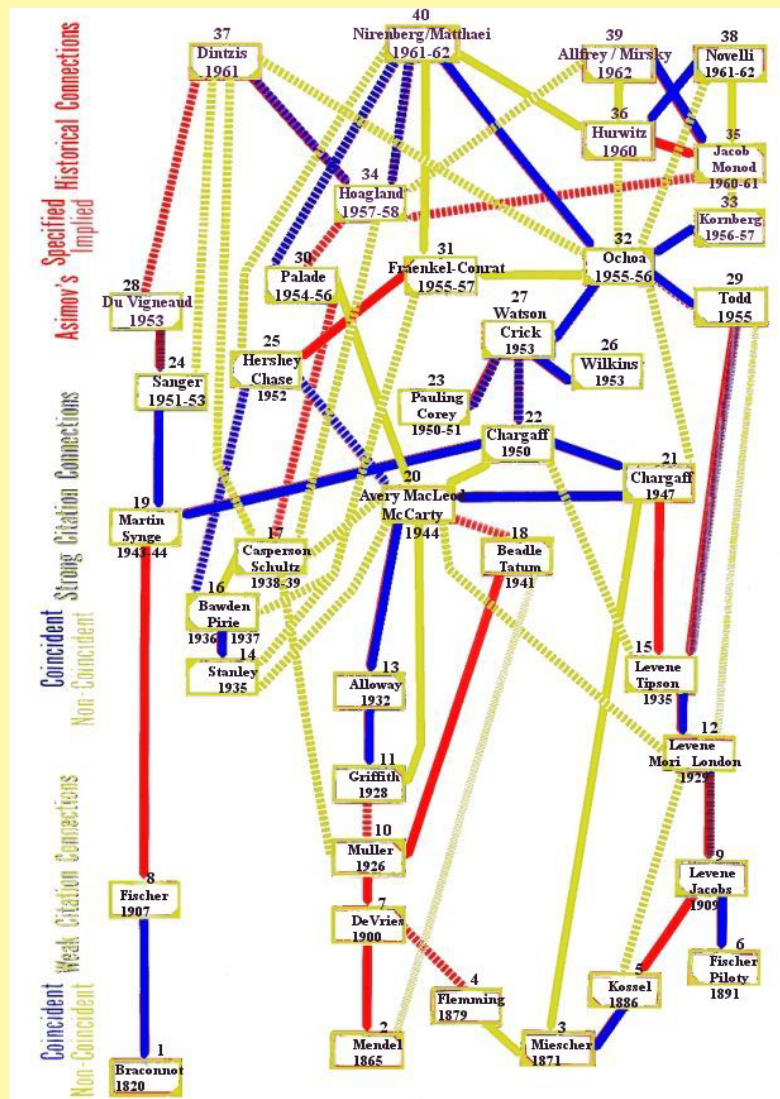
CPM (Critical Path Method): A project consists of tasks. Vertices of a project network represent states of the project and arcs represent tasks. Every project network is acyclic. For each task (u, v) its execution time $t(u, v)$ is known. A task can start only when all the preceding tasks are finished.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

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

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

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

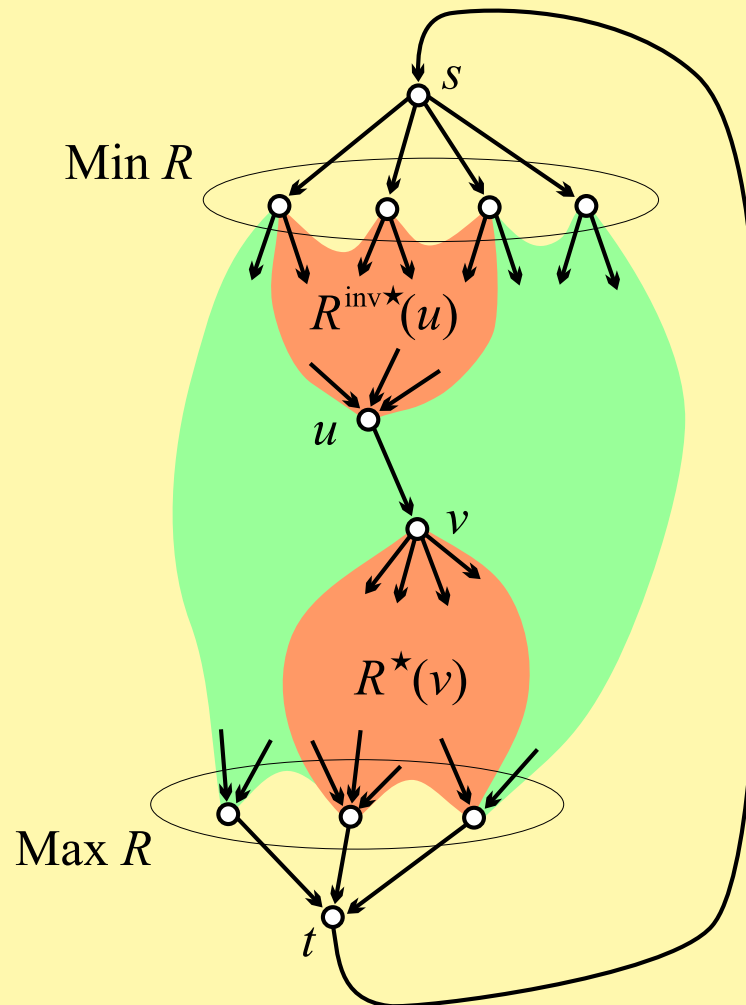
In a given set of units/vertices \mathcal{U} (articles, books, works, etc.) we introduce a *citing relation*/set of arcs $R \subseteq \mathcal{U} \times \mathcal{U}$

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

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

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 $s \notin \mathcal{U}$ and a common *sink* vertex $t \notin \mathcal{U}$. 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

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

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 \mathcal{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 \mathcal{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

$$t_c(u) = n^-(u) \cdot n^+(u)$$

$$t_l(u) = n_l^-(u) \cdot n_l^+(u)$$

$$t_p(u) = n_p^-(u) \cdot n_p^+(u)$$

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

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

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.

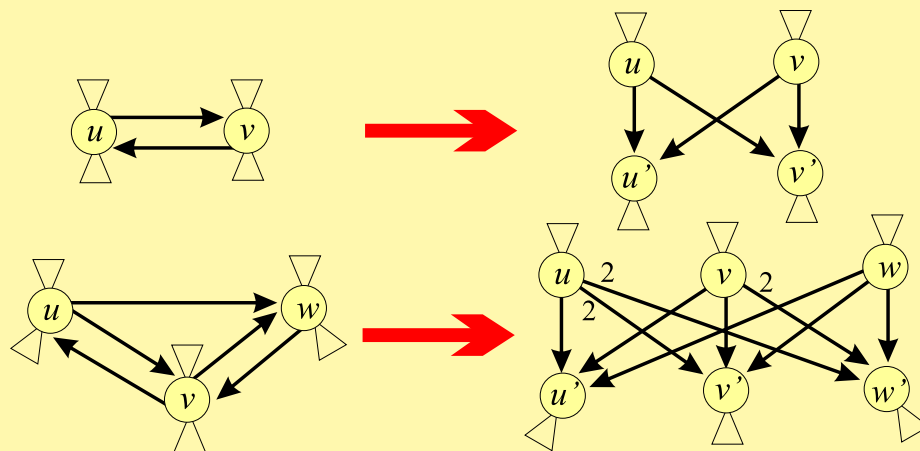
Nonacyclic citation networks

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.

Preprint transformation

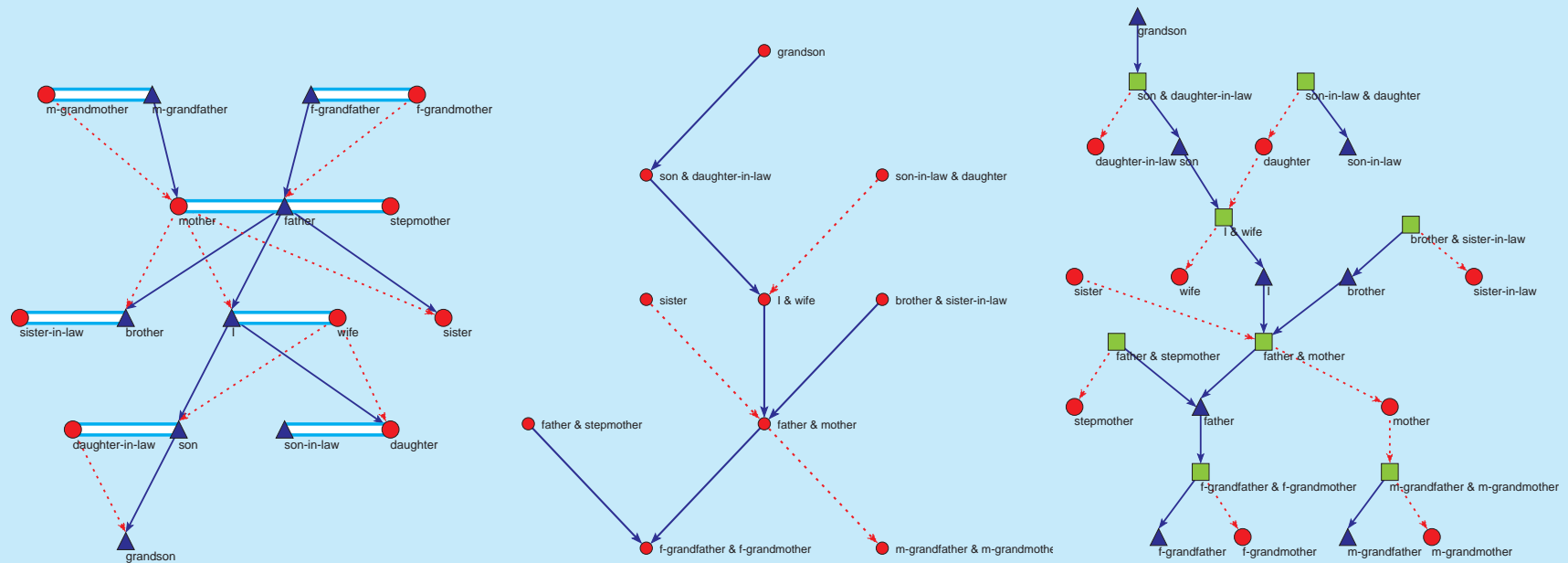


The *preprint transformation* is based on the following idea: Each paper from a strong component is duplicated with its 'preprint' version. The papers inside strong component cite preprints.

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.

Genealogies

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 vertices and lines 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 lines is of the same order as the number of vertices.

For a *regular Ore genealogy* we have (\mathcal{V} – vertices, \mathcal{A} – arcs, \mathcal{E} – edges):

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

p -graphs are almost trees – deviations from trees are caused by relinking marriages (\mathcal{V}_p , \mathcal{A}_p – vertices and arcs of p -graph):

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

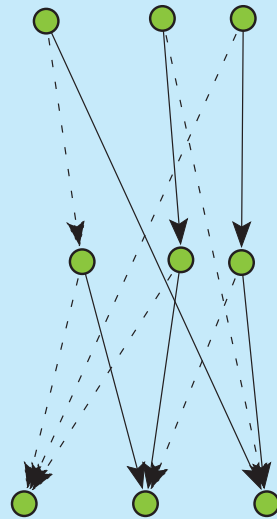
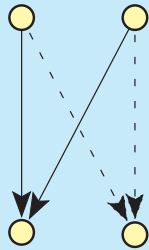
and for a bipartite p -graph, we have

$$|\mathcal{V}| \leq |\mathcal{V}_b| \leq \frac{3}{2}|\mathcal{V}|, \quad |\mathcal{A}_b| \leq 2|\mathcal{V}| + n_{mult}$$

Number of vertices and lines in Ore and p -graphs

data	$ \mathcal{V} $	$ \mathcal{E} $	$ \mathcal{A} $	$\frac{ \mathcal{L} }{ \mathcal{V} }$	$ \mathcal{V}_i $	n_{mult}	$ \mathcal{V}_p $	$ \mathcal{A}_p $	$\frac{ \mathcal{A}_p }{ \mathcal{V}_p }$
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 vertices in p -graph, m number of arcs, k number of weakly connected components, and M number of maximal vertices (vertices having output degree 0, $M \geq 1$).

The *relinking index* is defined as:

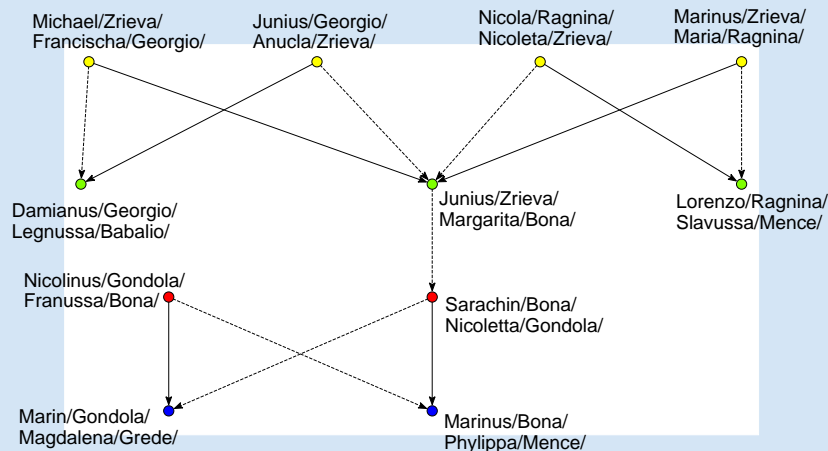
$$RI = \frac{k + m - n}{k + n - 2M}$$

For a trivial graph (having only one vertex) 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 molecules (carbon rings) and searching for relinking marriages in genealogies.



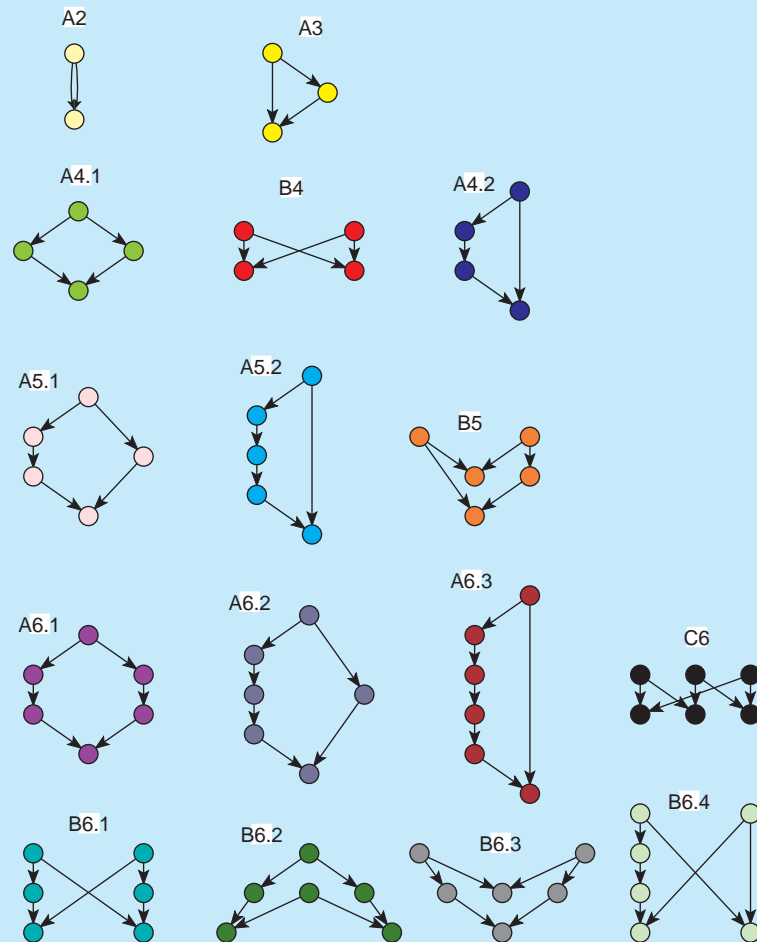
Three connected relinking marriages in the genealogy (represented as a p -graph) of ragusan noble families. A solid arc indicates the *_ is a son of _* relation, and a dotted arc indicates the *_ is a daughter of _* relation. In all three patterns a brother and a sister from one family found their partners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

...Pattern searching

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 molecule);
- 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.

Relinking patterns in p -graphs



All possible relinking marriages in p -graphs with 2 to 6 vertices. Patterns are labeled as follows:

- first character – number of first vertices: A – single, B – two, C – three.
- second character: number of vertices in pattern (2, 3, 4, 5, or 6).
- last character: identifier (if the two first characters are identical).

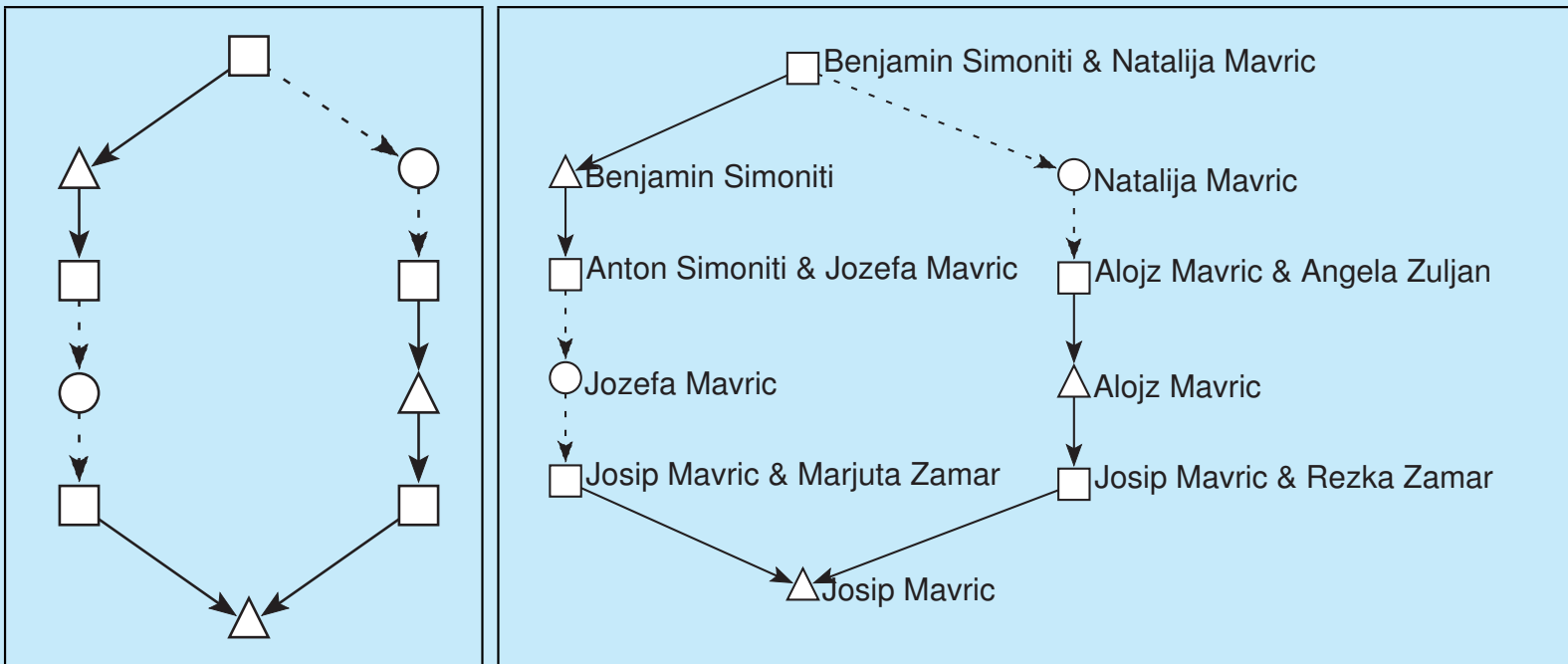
Patterns denoted by A are exactly the blood marriages. In every pattern the number of first vertices equals to the number of last vertices.

Frequencies normalized with number of couples in p -graph $\times 1000$.

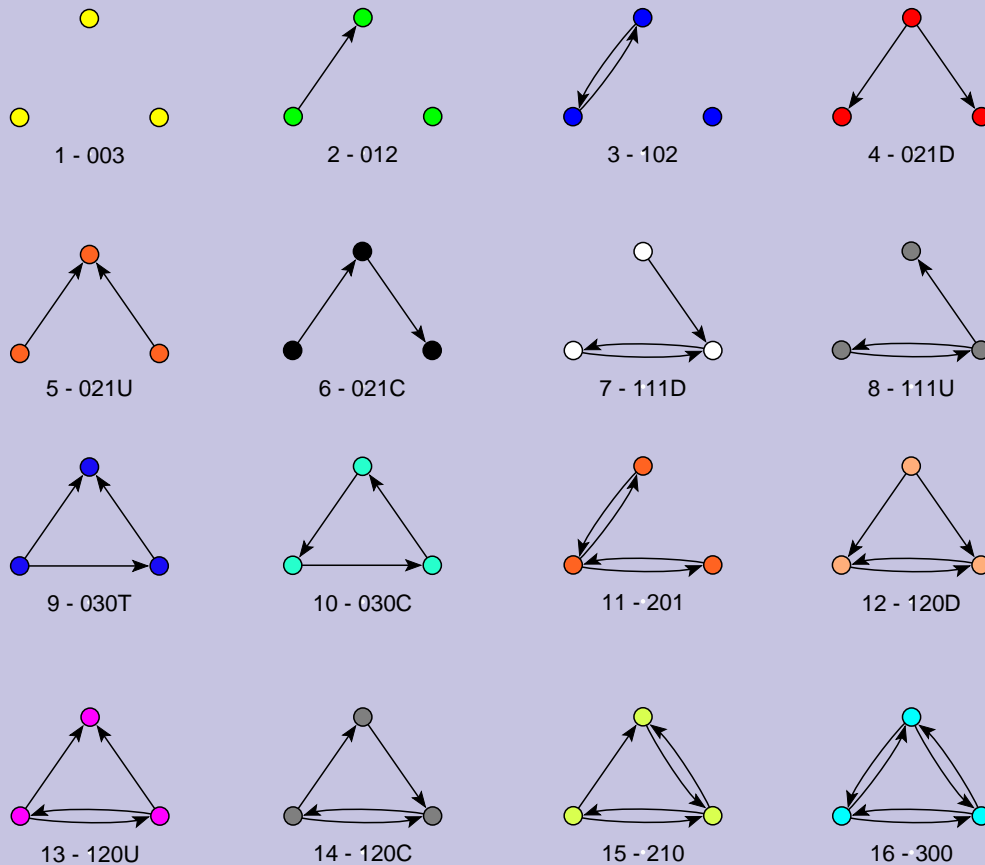
pattern	Loka	Silba	Ragusa	Turcs	Royal
A2	0.07	0.00	0.00	0.00	0.00
A3	0.07	0.00	0.00	0.00	2.64
A4.1	0.85	2.26	1.50	159.71	18.45
B4	3.82	11.28	10.49	98.28	6.15
A4.2	0.00	0.00	0.00	0.00	0.00
A5.1	0.64	3.16	2.00	36.86	11.42
A5.2	0.00	0.00	0.00	0.00	0.00
B5	1.34	4.96	23.48	46.68	7.03
A6.1	1.98	12.63	1.00	169.53	11.42
A6.2	0.00	0.90	0.00	0.00	0.88
A6.3	0.00	0.00	0.00	0.00	0.00
C6	0.71	5.41	9.49	36.86	4.39
B6.1	0.00	0.45	1.00	0.00	0.00
B6.2	1.91	17.59	31.47	130.22	10.54
B6.3	3.32	13.53	40.96	113.02	11.42
B6.4	0.00	0.00	2.50	7.37	0.00
Sum	14.70	72.17	123.88	798.53	84.36

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



Triads






Let $\mathcal{G} = (\mathcal{V}, R)$ 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 $\mathcal{N} = (\mathcal{U}, \mathcal{V}, A, w)$, where \mathcal{U} and \mathcal{V} are disjoint sets of vertices, A is the set of arcs with the initial vertex in the set \mathcal{U} and the terminal vertex in the set \mathcal{V} , and $w : A \rightarrow \mathbb{R}$ 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 $A \subseteq \mathcal{U} \times \mathcal{V}$.

A 2-mode network can be formally represented by rectangular matrix

$$\mathbf{A} = [a_{uv}]_{\mathcal{U} \times \mathcal{V}}.$$

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

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 $\mathcal{N}_1 = (\mathcal{U}, \mathcal{E}_1, w_1)$ or/and $\mathcal{N}_2 = (\mathcal{V}, \mathcal{E}_2, w_2)$, where \mathcal{E}_1 and w_1 are determined by the matrix $\mathbf{A}^{(1)} = \mathbf{A}\mathbf{A}^T$, $a_{uv}^{(1)} = \sum_{z \in \mathcal{V}} a_{uz} \cdot a_{zv}^T$. Evidently $a_{uv}^{(1)} = a_{vu}^{(1)}$. There is an edge $\{u, v\} \in \mathcal{E}_1$ in \mathcal{N}_1 iff $N(u) \cap N(v) \neq \emptyset$. Its weight is $w_1(u, v) = a_{uv}^{(1)}$.

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

The networks \mathcal{N}_1 and \mathcal{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

$$\text{Geo}_{uv} = \frac{w_{uv}}{\sqrt{w_{uu}w_{vv}}}$$

$$\text{GeoDeg}_{uv} = \frac{w_{uv}}{\sqrt{\text{deg}_u \text{deg}_v}}$$

$$\text{Input}_{uv} = \frac{w_{uv}}{w_{vv}}$$

$$\text{Output}_{uv} = \frac{w_{uv}}{w_{uu}}$$

$$\text{Min}_{uv} = \frac{w_{uv}}{\min(w_{uu}, w_{vv})}$$

$$\text{Max}_{uv} = \frac{w_{uv}}{\max(w_{uu}, w_{vv})}$$

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

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

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 magazines.

Reuters Terror News: **GeoDeg**, **MaxDir**, **MinDir**.

GeoDeg normalization of Reuters terror news network

