Photo: Stefan Ernst, ***Gartenkreuzspinne / Araneus diadematus***

Developing Pajek
Exploratory analysis of networks

Vladimir Batagelj
Andrej Mrvar

University of Ljubljana
Slovenia

Analyse des Données Relationnelles – EHESS-INED

INED, Paris, November 20, 2003

version: 13. december 2003 – 16 : 07

Outline

Networks

A *network* $\mathbf{N} = (\mathcal{V}, \mathcal{L}, \mathcal{T}, \mathcal{W})$ consists of:

- a *graph* $\mathbf{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of *vertices* and $\mathcal{L}$ is the set of *lines* (links, ties). Undirected lines are called *edges*, and directed lines *arcs*. $n = \mathrm{card}(\mathcal{V})$, $m = \mathrm{card}(\mathcal{L})$.
- $\mathcal{T}$ *vertex value functions*: $t : \mathcal{V} \rightarrow A$
- $\mathcal{W}$ *line value functions*: $w : \mathcal{L} \rightarrow B$

Large Networks

Large network – several thousands or millions of vertices.

Usually sparse $m \ll n^2$; typical: $m = O(n)$ or $m = O(n \log n)$.

Examples:

network	size	$n = \|V\|$	$m = \|L\|$	source
ODLIS dictionary	61K	2909	18419	ODLIS online
Citations SOM	168K	4470	12731	Garfield's collection
Molecula 1ATN	74K	5020	5128	Brookhaven PDB
Comput. geometry	140K	7343	11898	BiBTEX bibliographies
English words 2-8	520K	52652	89038	Knuth's English words
Internet traceroutes	1.7M	124651	207214	Internet Mapping Project
Franklin genealogy	12M	203909	195650	Roperld.com gedcoms
World-Wide-Web	3.6M	325729	1497135	Notre Dame Networks
Actors	3.9M	392400	1342595	Notre Dame Networks
US patents	82M	3774768	16522438	Nber
SI internet	38M	5547916	62259968	Najdi Si

Pajek

Pajek is a program, for Windows, for analysis and visualization of *large networks* having some ten or houndred of thousands of vertices.

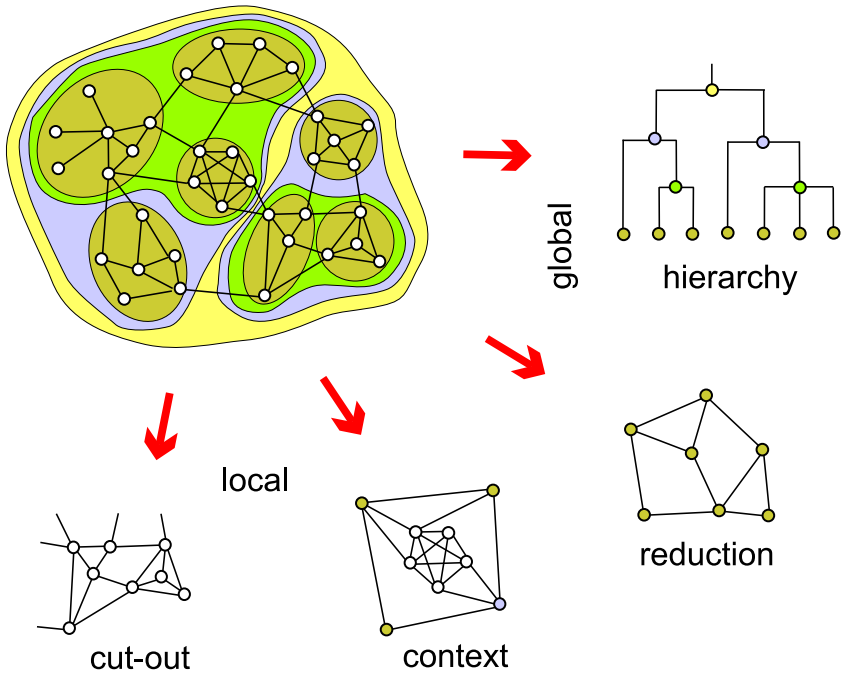
In Slovenian language *pajek* means spider.

The design of **Pajek** is based on experiences gained in development of graph data structure and algorithms libraries Graph and X-graph, collection of network analysis and visualization programs STRAN, and SGML-based graph description markup language NetML. We started the development of **Pajek** in November 1996. **Pajek** is implemented in Delphi. Some procedures were contributed by Matjaž Zaveršnik.

The latest version of **Pajek** is freely available, for noncommercial use, at its home page:

http://vlado.fmf.uni-lj.si/pub/networks/pajek/

Main design goals



The main goals in the design of **Pajek** are:

- to support abstraction by (recursive) *decomposition* of a large network into several smaller networks that can be treated further using more sophisticated methods;
- to provide the user with some powerful *visualization* tools;
- to implement a selection of efficient *subquadratic* algorithms for analysis of large networks.

With **Pajek** we can: *find* clusters (components, neighbourhoods of 'important' vertices, cores, etc.) in a network, *extract* vertices that belong to the same clusters and *show* them separately, possibly with the parts of the context (detailed local view), *shrink* vertices in clusters and show relations among clusters (global view).

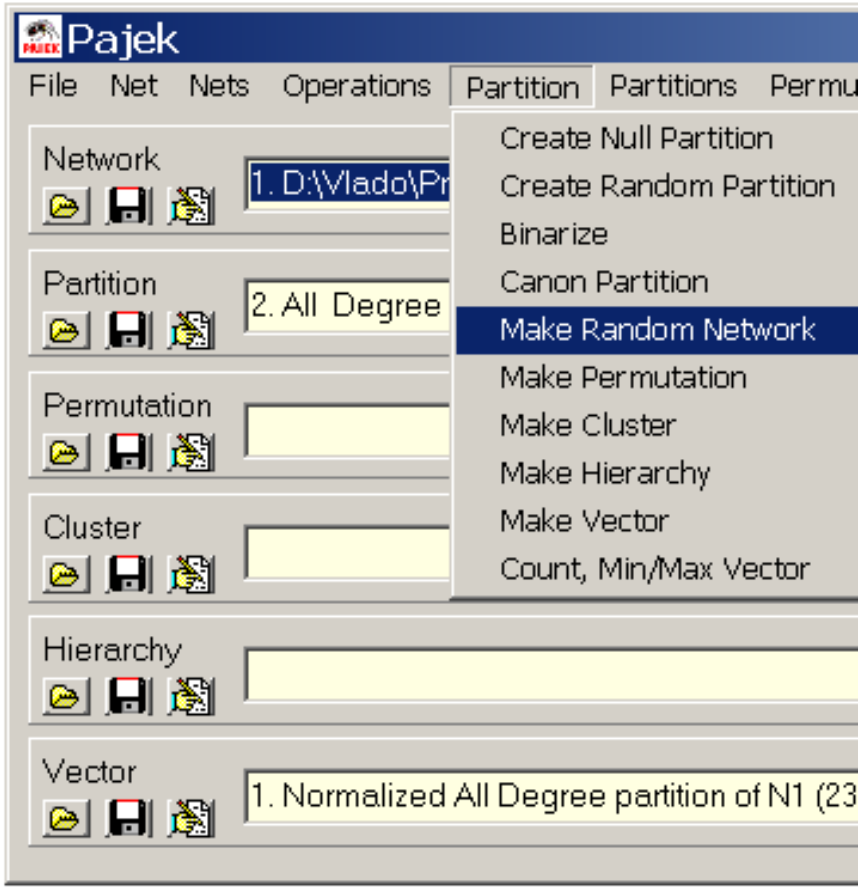
Types of networks

Besides ordinary (directed, undirected, mixed) networks **Pajek** supports also:

- *2-mode networks*, bipartite (valued) graphs – networks between two disjoint sets of vertices. Examples of such networks are: (authors, papers, *cites the paper*), (authors, papers, *is the (co)author of the paper*), (people, events, *was present at*), (people, institutions, *is member of*), (articles, shoping lists, *is on the list*).
- *temporal networks*, dynamic graphs – networks changing over time.

Pajek's data types

In **Pajek** analysis and visualization are performed using 6 data types:



- *network* (graph),
- *partition* (nominal or ordinal properties of vertices),
- *vector* (numerical properties of vertices),
- *cluster* (subset of vertices),
- *permutation* (reordering of vertices, ordinal properties), and
- *hierarchy* (general tree structure on vertices).

We intend to extend this list with a support of *multiple networks* and *partitions of lines*.

...Pajek's data types

The power of **Pajek** is based on several transformations that support different transitions among these data structures. Also the menu structure of the main **Pajek**'s window is based on them. **Pajek**'s main window uses a 'calculator' paradigm with list-accumulator for each data type. The operations are performed on the currently active (selected) data and are also returning the results through accumulators.

The values of vectors can be used to determine several elements of network display such as: X, Y, Z coordinates and the size of the vertex shape. The partition can be graphically represented by the color and shape of vertices. Also the values of edges can be represented by the thickness and/or color.

Algorithms

To support the design goals we implemented several algorithms known from the literature, but for some tasks new, efficient algorithms, suitable to deal with large networks, had to be developed. They mainly provide different ways to identify interesting substructures in a given network.

To extend the range of **Pajek**, on very large networks most basic operations work in-place (destroying the input network).

Standard algorithms

In **Pajek** several known efficient algorithms are implemented, like:

- *simplifications and transformations*: deleting loops, multiple edges, transforming arcs to edges etc.;
- *components*: strong, weak, biconnected, symmetric;
- *decompositions*: symmetric-acyclic, hierarchical clustering;
- *paths*: shortest path(s), all paths between two vertices;
- *flows*: maximum flow between two vertices;
- *neighborhood*: k-neighbours;
- *CPM* – critical paths;

Standard algorithms

- *social networks algorithms*: centrality measures, hubs and authorities, measures of prestige, brokerage roles, structural holes, diffusion partitions;
- *measures of dependencies among partitions / vectors*: Cramer's V, Spearman rank correlation coefficient, Pearson correlation coefficient, Rajski coefficient;
- *extracting* subnetwork;
- *shrinking* clusters in network (generalized blockmodeling);
- *reordering*: topological ordering, Richards's numbering, Murtagh's seriation and clumping algorithms, depth/breadth first search;

Special algorithms

- *islands*:
- *citation weights*:
- *cores and generalized cores*:
- *pattern searching*:
- *triads*:
- *triangular connectivities*:
- *generating large random networks*;
- *2-mode networks* normalizations;
- *generalized blockmodeling*;

Algorithms for small networks

Pajek contains also some data analysis procedures which have higher order time complexities and can be therefore used only on smaller networks, or selected parts of large networks: hierarchical clustering, generalized blockmodeling, partitioning signed graphs, TSP (Traveling Salesman Problem), computing geodesics matrices, etc.

The procedures are available through the main window menus. Frequently used sequences of operations can be defined as *macro*s. This allows also the adaptations of **Pajek** to groups of users from different areas (social networks, chemistry, genealogy, computer science, mathematics...) for specific tasks.

Layout Algorithms and Layout Features

Special emphasis is given in **Pajek** to automatic generation of network layouts. Several standard algorithms for automatic graph drawing are implemented: *spring embedders* (Kamada-Kawai and Fruchterman-Reingold), layouts determined by *eigenvectors* (Lanczos algorithm), drawing in *layers* (genealogies and other acyclic structures), *fish-eye* views and *block* (matrix) representation.

These algorithms were modified and extended to enable additional options: drawing with constraints (optimization of the selected part of the network, fixing some vertices to predefined positions, using values of edges as similarities or dissimilarities), drawing in 3D space. **Pajek** also provides tools for manual editing of graph layout.

. . . Layout Algorithms and Layout Features

Properties of vertices/edges (given as data or computed) can be represented using colors, sizes and/or shapes of vertices/edges.

Pajek supports also drawing sequences of networks in its Draw window, and exports sequences of networks in suitable formats that can be examined with special 2D or 3D viewers (e.g., SVG and Mage). Pictures in SVG can be further controled using support written in Javascript.

Interfaces

Pajek supports also some non-native input formats: UCINET DL files; Vega graph files; chemical MDLMOL and BS; and genealogical GEDCOM.

The layouts can be exported in the following output graphic formats that can be examined by special 2D and 3D viewers: Encapsulated PostScript (EPS), Scalable Vector Graphics (SVG), VRML, MDLMOL/ chime, and Kinemages (Mage).

The main window menu `Tools` provides export of **Pajek**'s data to statistical program R. In the `Tools` menu, the user can prepare calls to her/his favorite viewers and other tools. It is also possible to run **Pajek** (+macros) from other programs (R, Ucinet, and others).

Applications

There exist several sources of large networks that are already in machine-readable form. **Pajek** provides tools for analysis and visualization of such networks and is applied by researchers in different areas: social network analysis, chemistry (organic molecule), biomedical/genomics research, genealogies, Internet networks, collaboration networks, diffusion networks (AIDS, news), analysis of texts, data-mining (2-mode networks), etc. Although it was developed primarily for analysis of large networks it is often used also for, especially visualization of, small networks.

Pajek is also used at several universities as a support in courses on network analysis. Together with Wouter de Nooy from University of Rotterdam we wrote a course book *Exploratory Social Network Analysis With* **Pajek**.

Selected problems and algorithms

To support the design goals we implemented several algorithms known from the literature, but for some tasks new, efficient algorithms, suitable to deal with large networks, had to be developed. In the following we present some of our contributions in this direction.

Cuts

The *line-cut* in network $\mathbf{N} = (V, L, w)$, $w : L \rightarrow \mathbb{R}$ at selected level t is a subnetwork $\mathbf{N}(t) = (V(L'), L', w)$ induced by the set of lines

$$L' = \{e \in L : w(e) \geq t\}$$

where $V(L')$ is the set of all endvertices of the lines from L'.

The *vertex-cut* in network $\mathbf{N} = (V, L, p)$, $p : V \rightarrow \mathbb{R}$ at selected level t is a subnetwork $\mathbf{N}(t) = (V', L(V'), p)$, induced by the set

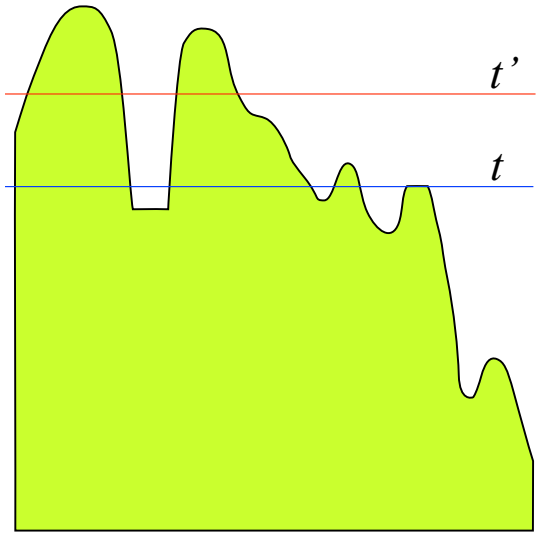
$$V' = \{v \in V : p(v) \geq t\}$$

where $L(V')$ is the set of lines from L that have both endvertices in V'.

To obtain interesting themes we consider only components of size at least k.

The values of thresholds t and k are determined by inspecting the distribution of weights/values and the distribution of component sizes.

Islands (with M. Zaveršnik)



A set of vertices $C \subseteq V$ is a *vertex island* in network $\mathbf{N} = (V, L, p)$, $p : V \rightarrow \mathbb{R}$ iff it induces a connected subgraph and the vertices from the island are 'higher' than the neighboring vertices

$$\max_{u \in N(C)} p(u) < \min_{v \in c} p(v)$$

A set of vertices $C \subseteq V$ is a *line island* in network $\mathbf{N} = (V, L, w)$, $w : L \rightarrow \mathbb{R}$ iff it induces a connected subgraph and the lines inside the island are 'stronger related' among them than with the neighboring vertices – in $\mathbf{N}$ there exists a spanning tree T over C such that

$$\max_{(u,v) \in L, u \notin C, v \in C} w(u, v) < \min_{(u,v) \in T} w(u, v)$$

. . . islands

We can also define *simple* islands with a single 'peak'.

Usually we are interested only in networks of size between k and K.

Island, also those with restricted sizes $k..K$, determine hierarhies over V.

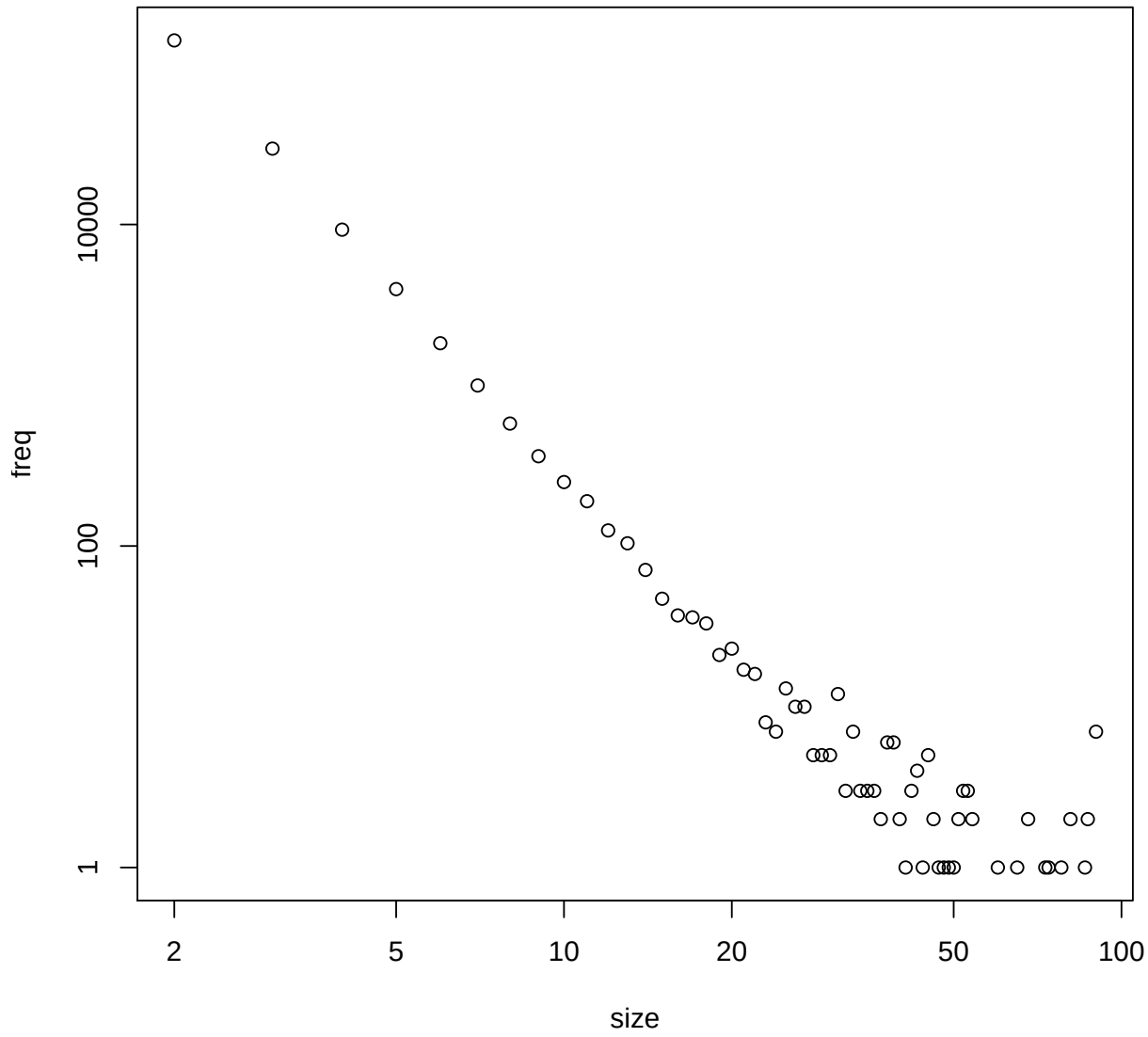
There exist efficient algorithms to determine the islands.

Example – US patents

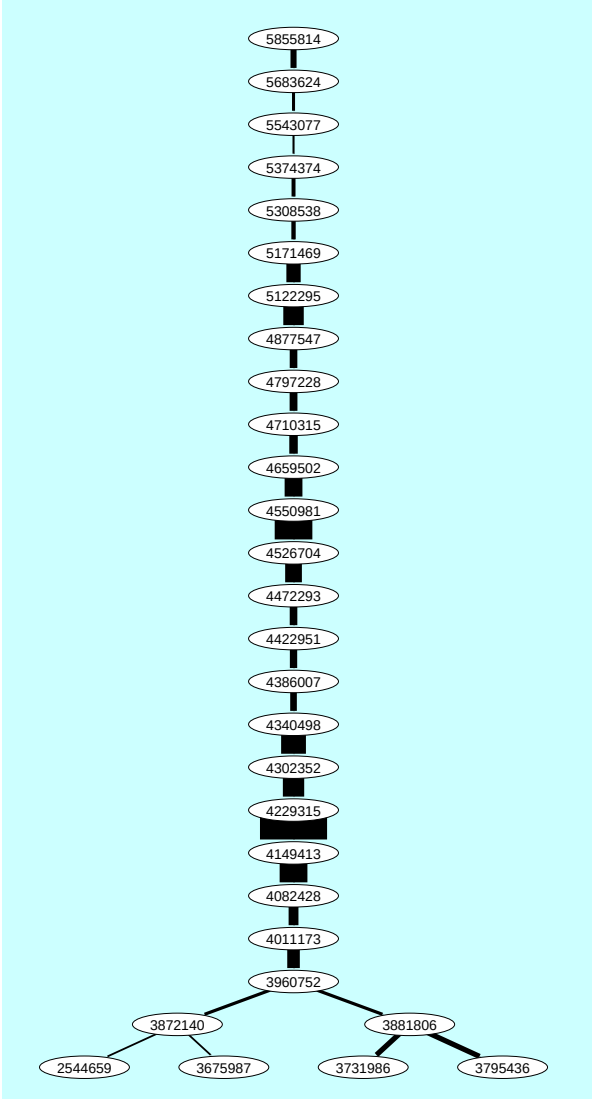
As an example, let us look at Nber network of US Patents. It has 3774768 vertices and 16522438 arcs (1 loop). We computed SPC weights in it and determined all (2,90)-islands. The reduced network has 470137 vertices, 307472 arcs and for different k: C_2 =187610, C_5 =8859,C_{30} =101, C_{50} =30 islands. Rolex

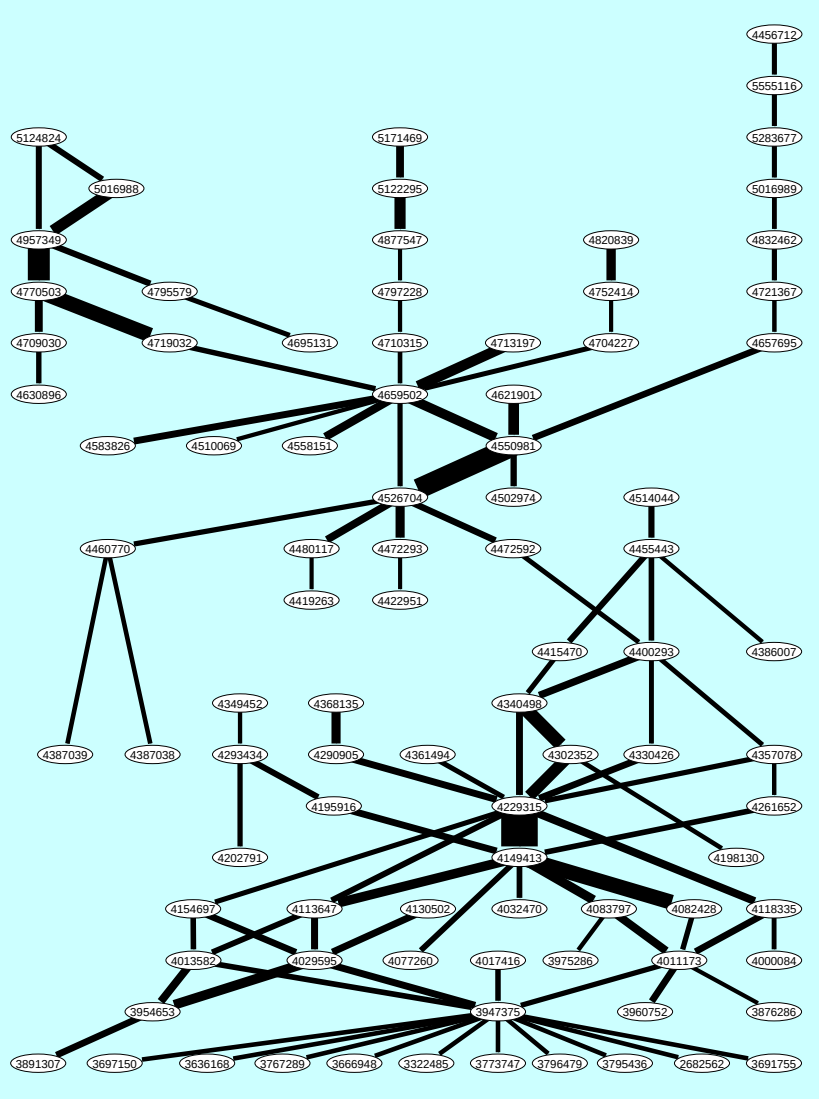
```
  [1]     0 139793  29670 9288 3966 1827 997 578 362 250
[11]  190    125    104   71   47   37  36  33  21  23
[21]   17     16      8    7   13   10  10   5   5   5
[31]   12      3      7    3    3    3   2   6   6   2
[41]    1      3      4    1    5    2   1   1   1   1
[51]    2      3      3    2    0    0   0   0   0   1
[61]    0      0      0    0    1    0   0   2   0   0
[71]    0      0      1    1    0    0   0   1   0   0
[81]    2      0      0    0    0    1   2   0   0   7
```

Island size distribution



Main path and main island of Patents





Liquid crystal display

Table 1: Patents on the liquid-crystal display

patent	date	author(s) and title
2544659	Mar 13, 1951	Dreyer. Dichroic light-polarizing sheet and the like and the formation and use thereof
2682562	Jun 29, 1954	Wender, et al. Reduction of aromatic carbinols
3322485	May 30, 1967	Williams. Electro-optical elements utilazing an organic nematic compound
3636168	Jan 18, 1972	Josephson. Preparation of polynuclear aromatic compounds
3666948	May 30, 1972	Mechlowitz, et al. Liquid crystal termal imaging system having an undisturbed image on a disturbed background
3675987	Jul 11, 1972	Rafuse. Liquid crystal compositions and devices
3691755	Sep 19, 1972	Girard. Clock with digital display
3697150	Oct 10, 1972	Wysochi. Electro-optic systems in which an electrophoretic-like or dipolar material is dispersed throughout a liquid crystal to reduce the turn-off time
3731986	May 8, 1973	Fergason. Display devices utilizing liquid crystal light modulation
3767289	Oct 23, 1973	Aviram, et al. Class of stable trans-stilbene compounds, some displaying nematic mesophases at or near room temperature and others in a range up to 100°C
3773747	Nov 20, 1973	Steinstrasser. Substituted azoxy benzene compounds
3795436	Mar 5, 1974	Boller, et al. Nematogenic material which exhibit the Kerr effect at isotropic temperatures
3796479	Mar 12, 1974	Helfrich, et al. Electro-optical light-modulation cell utilizing a nematogenic material which exhibits the Kerr effect at isotropic temperatures
3872140	Mar 18, 1975	Klanderman, et al. Liquid crystalline compositions and method
3876286	Apr 8, 1975	Deutscher, et al. Use of nematic liquid crystalline substances
3881806	May 6, 1975	Suzuki. Electro-optical display device
3891307	Jun 24, 1975	Tsukamoto, et al. Phase control of the voltages applied to opposite electrodes for a cholesteric to nematic phase transition display
3947375	Mar 30, 1976	Gray, et al. Liquid crystal materials and devices
3954653	May 4, 1976	Yamazaki. Liquid crystal composition having high dielectric anisotropy and display device incorporating same
3960752	Jun 1, 1976	Klanderman, et al. Liquid crystal compositions
3975286	Aug 17, 1976	Oh. Low voltage actuated field effect liquid crystals compositions and method of synthesis
4000084	Dec 28, 1976	Hsieh, et al. Liquid crystal mixtures for electro-optical display devices
4011173	Mar 8, 1977	Steinstrasser. Modified nematic mixtures with positive dielectric anisotropy
4013582	Mar 22, 1977	Gavrilovic. Liquid crystal compounds and electro-optic devices incorporating them
4017416	Apr 12, 1977	Inukai, et al. P-cyanophenyl 4-alkyl-4'-biphenylcarboxylate, method for preparing same and liquid crystal compositions using same
4029595	Jun 14, 1977	Ross, et al. Novel liquid crystal compounds and electro-optic devices incorporating them
4032470	Jun 28, 1977	Bloom, et al. Electro-optic device
4077260	Mar 7, 1978	Gray, et al. Optically active cyano-biphenyl compounds and liquid crystal materials containing them
4082428	Apr 4, 1978	Hsu. Liquid crystal composition and method

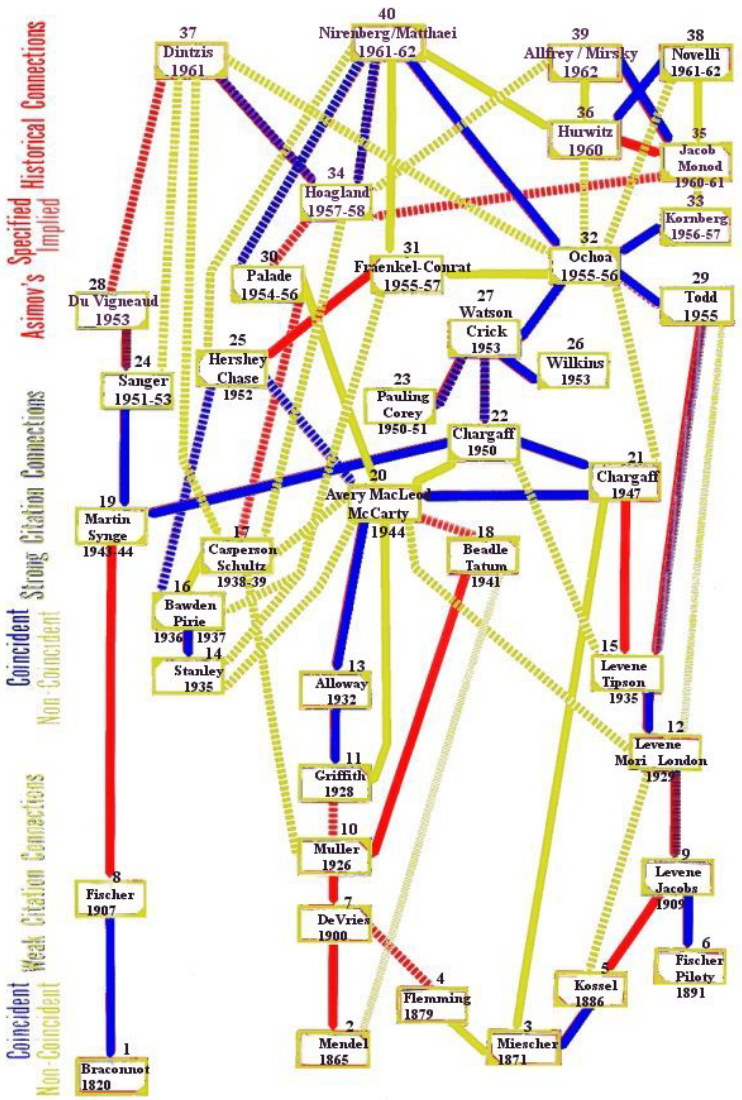
Table 2: Patents on the liquid-crystal display

patent	date	author(s) and title
4083797	Apr 11, 1978	Oh. Nematic liquid crystal compositions
4113647	Sep 12, 1978	Coates, et al. Liquid crystalline materials
4118335	Oct 3, 1978	Krause, et al. Liquid crystalline materials of reduced viscosity
4130502	Dec 19, 1978	Eidenschink, et al. Liquid crystalline cyclohexane derivatives
4149413	Apr 17, 1979	Gray, et al. Optically active liquid crystal mixtures and liquid crystal devices containing them
4154697	May 15, 1979	Eidenschink, et al. Liquid crystalline hexahydroterphenyl derivatives
4195916	Apr 1, 1980	Coates, et al. Liquid crystal compounds
4198130	Apr 15, 1980	Boller, et al. Liquid crystal mixtures
4202791	May 13, 1980	Sato, et al. Nematic liquid crystalline materials
4229315	Oct 21, 1980	Krause, et al. Liquid crystalline cyclohexane derivatives
4261652	Apr 14, 1981	Gray, et al. Liquid crystal compounds and materials and devices containing them
4290905	Sep 22, 1981	Kanbe. Ester compound
4293434	Oct 6, 1981	Deutscher, et al. Liquid crystal compounds
4302352	Nov 24, 1981	Eidenschink, et al. Fluorophenylcyclohexanes, the preparation thereof and their use as components of liquid crystal dielectrics
4330426	May 18, 1982	Eidenschink, et al. Cyclohexylbiphenyls, their preparation and use in dielectrics and electrooptical display elements
4340498	Jul 20, 1982	Sugimori. Halogenated ester derivatives
4349452	Sep 14, 1982	Osman, et al. Cyclohexylcyclohexanoates
4357078	Nov 2, 1982	Carr, et al. Liquid crystal compounds containing an alicyclic ring and exhibiting a low dielectric anisotropy and liquid crystal materials and devices incorporating such compounds
4361494	Nov 30, 1982	Osman, et al. Anisotropic cyclohexyl cyclohexylmethyl ethers
4368135	Jan 11, 1983	Osman. Anisotropic compounds with negative or positive DC-anisotropy and low optical anisotropy
4386007	May 31, 1983	Krause, et al. Liquid crystalline naphthalene derivatives
4387038	Jun 7, 1983	Fukui, et al. 4-(Trans-4'-alkylcyclohexyl) benzoic acid 4"'-cyano-4"-biphenylyl esters
4387039	Jun 7, 1983	Sugimori, et al. Trans-4-(trans-4'-alkylcyclohexyl)-cyclohexane carboxylic acid 4"'-cyanobiphenyl ester
4400293	Aug 23, 1983	Romer, et al. Liquid crystalline cyclohexylphenyl derivatives
4415470	Nov 15, 1983	Eidenschink, et al. Liquid crystalline fluorine-containing cyclohexylbiphenyls and dielectrics and electro-optical display elements based thereon
4419263	Dec 6, 1983	Praefcke, et al. Liquid crystalline cyclohexylcarbonitrile derivatives
4422951	Dec 27, 1983	Sugimori, et al. Liquid crystal benzene derivatives
4455443	Jun 19, 1984	Takatsu, et al. Nematic halogen Compound
4456712	Jun 26, 1984	Christie, et al. Bismaleimide triazine composition
4460770	Jul 17, 1984	Petrzilka, et al. Liquid crystal mixture
4472293	Sep 18, 1984	Sugimori, et al. High temperature liquid crystal substances of four rings and liquid crystal compositions containing the same
4472592	Sep 18, 1984	Takatsu, et al. Nematic liquid crystalline compounds
4480117	Oct 30, 1984	Takatsu, et al. Nematic liquid crystalline compounds
4502974	Mar 5, 1985	Sugimori, et al. High temperature liquid-crystalline ester compounds
4510069	Apr 9, 1985	Eidenschink, et al. Cyclohexane derivatives

Table 3: Patents on the liquid-crystal display

patent	date	author(s) and title
4514044	Apr 30, 1985	Gunjima, et al. 1-(Trans-4-alkylcyclohexyl)-2-(trans-4'-(p-substituted phenyl) cyclohexyl)ethane and liquid crystal mixture
4526704	Jul 2, 1985	Petrzilka, et al. Multiring liquid crystal esters
4550981	Nov 5, 1985	Petrzilka, et al. Liquid crystalline esters and mixtures
4558151	Dec 10, 1985	Takatsu, et al. Nematic liquid crystalline compounds
4583826	Apr 22, 1986	Petrzilka, et al. Phenylethanes
4621901	Nov 11, 1986	Petrzilka, et al. Novel liquid crystal mixtures
4630896	Dec 23, 1986	Petrzilka, et al. Benzonitriles
4657695	Apr 14, 1987	Saito, et al. Substituted pyridazines
4659502	Apr 21, 1987	Fearon, et al. Ethane derivatives
4695131	Sep 22, 1987	Balkwill, et al. Disubstituted ethanes and their use in liquid crystal materials and devices
4704227	Nov 3, 1987	Krause, et al. Liquid crystal compounds
4709030	Nov 24, 1987	Petrzilka, et al. Novel liquid crystal mixtures
4710315	Dec 1, 1987	Schad, et al. Anisotropic compounds and liquid crystal mixtures therewith
4713197	Dec 15, 1987	Eidenschink, et al. Nitrogen-containing heterocyclic compounds
4719032	Jan 12, 1988	Wachtler, et al. Cyclohexane derivatives
4721367	Jan 26, 1988	Yoshinaga, et al. Liquid crystal device
4752414	Jun 21, 1988	Eidenschink, et al. Nitrogen-containing heterocyclic compounds
4770503	Sep 13, 1988	Buchecker, et al. Liquid crystalline compounds
4795579	Jan 3, 1989	Vauchier, et al. 2,2'-difluoro-4-alkoxy-4'-hydroxydiphenyls and their derivatives, their production process and their use in liquid crystal display devices
4797228	Jan 10, 1989	Goto, et al. Cyclohexane derivative and liquid crystal composition containing same
4820839	Apr 11, 1989	Krause, et al. Nitrogen-containing heterocyclic esters
4832462	May 23, 1989	Clark, et al. Liquid crystal devices
4877547	Oct 31, 1989	Weber, et al. Liquid crystal display element
4957349	Sep 18, 1990	Clerc, et al. Active matrix screen for the color display of television pictures, control system and process for producing said screen
5016988	May 21, 1991	Iimura. Liquid crystal display device with a birefringent compensator
5016989	May 21, 1991	Okada. Liquid crystal element with improved contrast and brightness
5122295	Jun 16, 1992	Weber, et al. Matrix liquid crystal display
5124824	Jun 23, 1992	Kozaki, et al. Liquid crystal display device comprising a retardation compensation layer having a maximum principal refractive index in the thickness direction
5171469	Dec 15, 1992	Hittich, et al. Liquid-crystal matrix display
5283677	Feb 1, 1994	Sagawa, et al. Liquid crystal display with ground regions between terminal groups
5308538	May 3, 1994	Weber, et al. Supertwist liquid-crystal display
5374374	Dec 20, 1994	Weber, et al. Supertwist liquid-crystal display
5543077	Aug 6, 1996	Rieger, et al. Nematic liquid-crystal composition
5555116	Sep 10, 1996	Ishikawa, et al. Liquid crystal display having adjacent electrode terminals set equal in length
5683624	Nov 4, 1997	Sekiguchi, et al. Liquid crystal composition
5855814	Jan 5, 1999	Matsui, et al. Liquid crystal compositions and liquid crystal display elements

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

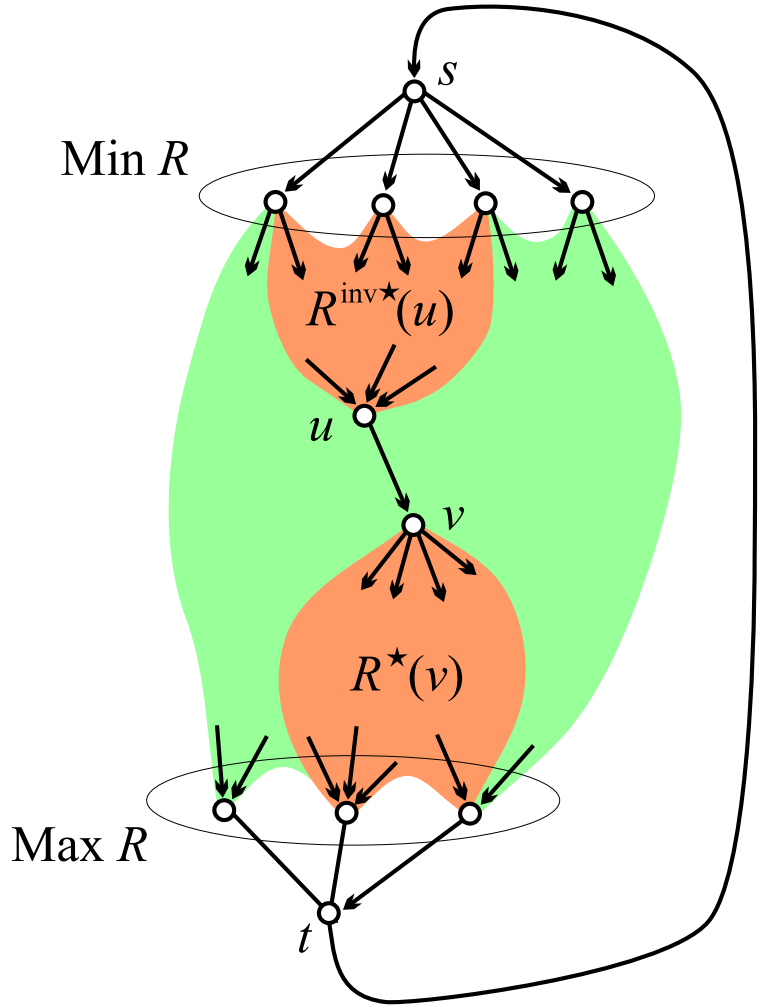
In a given set of units/vertices $\mathbf{U}$ (articles, books, works, etc.) we introduce a *citing relation*/set of arcs $R \subseteq \mathbf{U} \times \mathbf{U}$

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

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 **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.

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

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

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.

SOM main subnetwork

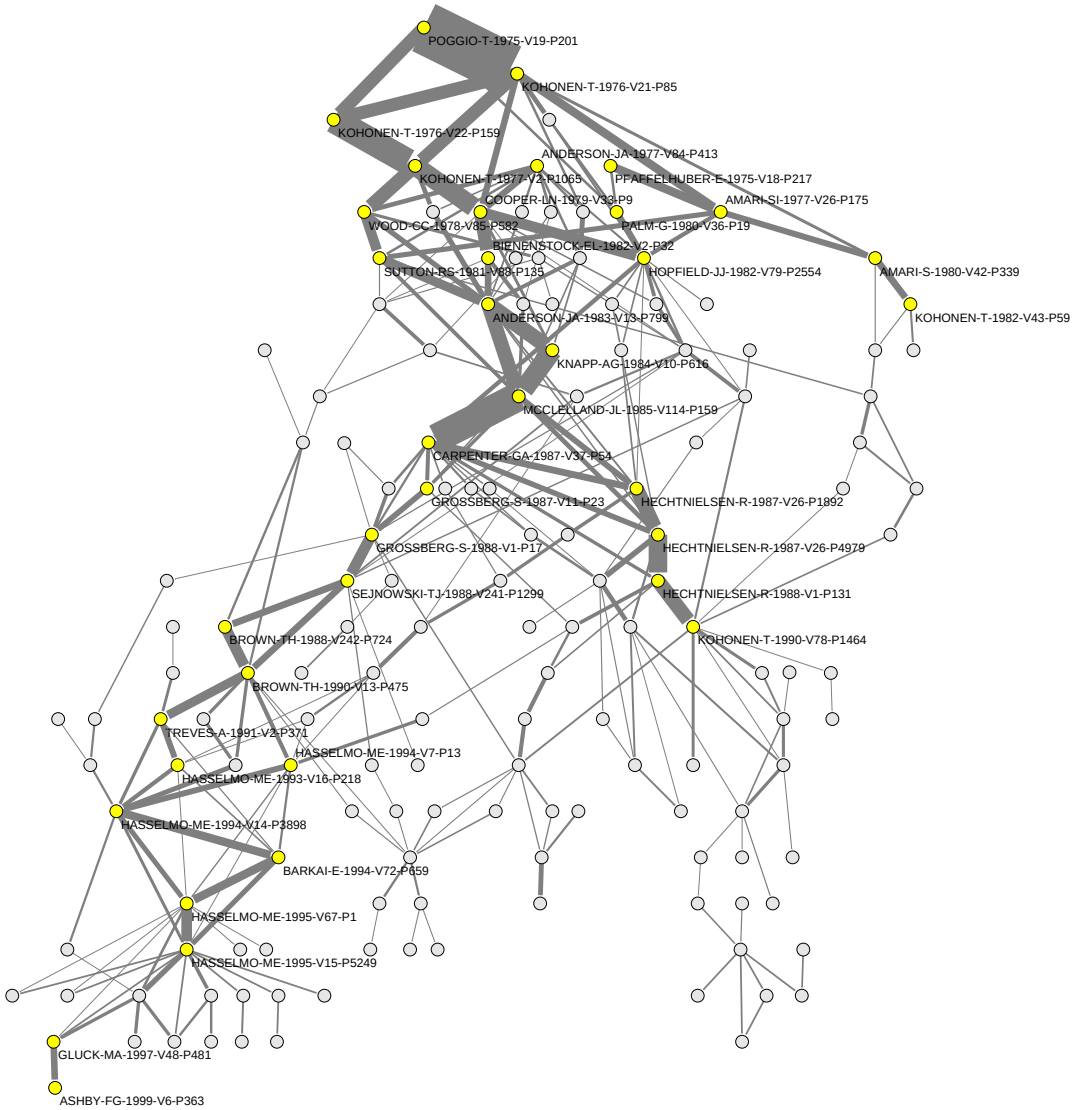
Consider citation network ($n = 4470$, $m = 12731$) on SOM (*self-organizing maps*) literature.

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.

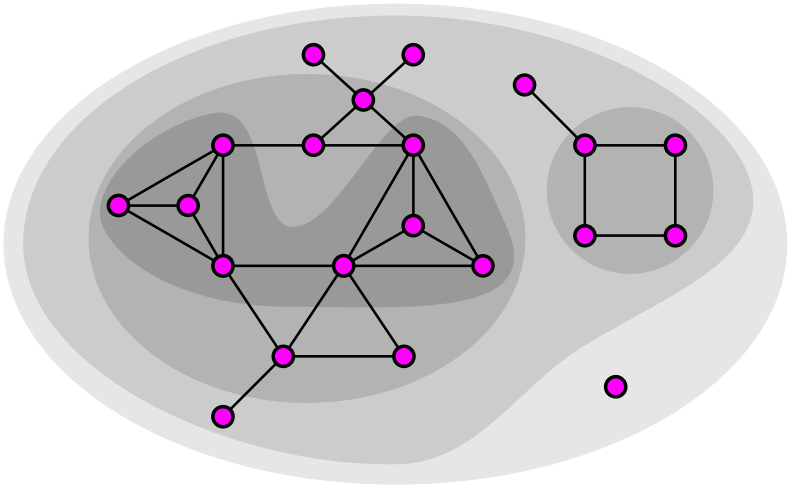
We label only the 'important' vertices – endpoints of arcs with weight at least 0.05.

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



Cores and generalized cores (with M. Zaveršnik)



The notion of core was introduced by Seidman in 1983. Let $\mathbf{G} = (V, E)$ be a graph. A subgraph $H = (W, E|W)$ induced by the set W is a k-*core* or a *core of order* k iff $\forall v \in W : \deg_H(v) \geq k$, and H is a maximal subgraph with this property. The core of maximum order is also called the *main* core.

The *core number* of vertex v is the highest order of a core that contains this vertex. The degree $\deg(v)$ can be: in-degree, out-degree, in-degree + out-degree, etc., determining different types of cores.

Properties of cores

From the figure, representing 0, 1, 2 and 3 core, we can see the following properties of cores:

- The cores are nested: $i < j \implies H_j \subseteq H_i$
- Cores are not necessarily connected subgraphs.

Our algorithm for determining the cores hierarchy is based on the following property:

> If from a given graph $\mathbf{G} = (V, E)$ we recursively delete all vertices, and edges incident with them, of degree less than k, the remaining graph is the k-core.

Core Numbers Algorithm

Input : Graph $G = (V, E)$ represented by lists of neighbors

Output : Table $core[V]$ with core number for each vertex

Compute the $degree$s of vertices

Order the set of vertices V in increasing order of their degrees

for *each* $v \in V$ *in the order* **do**
    Set $core[v] = degree[v]$
    for *each* $u \in \mathrm{adj}(v)$ **do**
        if $degree[u] > degree[v]$ **then**
            Set $degree[u] = degree[u] - 1$
            Reorder V accordingly
        end
    end
end

. . . Core Numbers Algorithm

In the refinements of the algorithm we have to provide efficient implementations of sorting the $degree$s and their reordering. Since the values of degrees are in the range $0..n-1$ we can order them in $O(n)$ using a variant of bin sort; and the update of the ordering can be done in a constant time.

The cores, because they can be determined very efficiently, are one among few concepts that provide us with meaningful decompositions of large networks. We expect that different approaches to the analysis of large networks can be built on this basis. For example: we get the following bound on the chromatic number of a given graph $\mathbf{G}$

$$\chi(\mathbf{G}) \leq 1 + \mathrm{core}(\mathbf{G})$$

Cores can also be used to localize the search for interesting subnetworks in large networks since: if it exists, a k-component is contained in a k-core; and a k-clique is contained in a k-core.

Generalized cores

The notion of core can be generalized to networks. Let $\mathbf{N} = (V, E, w)$ be a network, where $\mathbf{G} = (V, E)$ is a graph and $w : E \rightarrow \mathbb{R}$ is a function assigning values to edges. A *vertex property function* on $\mathbf{N}$, or a *p-function* for short, is a function $p(v, U)$, $v \in V$, $U \subseteq V$ with real values. Let $\mathrm{adj}_U(v) = \mathrm{adj}(v) \cap U$. Besides degrees, here are some examples of p-functions:

$$p_S(v,U) = \sum_{u \in \mathrm{adj}_U(v)} w(v,u), \text{ where } w : E \rightarrow \mathbb{R}_0^+$$

$$p_M(v,U) = \max_{u \in \mathrm{adj}_U(v)} w(v,u), \text{ where } w : E \rightarrow \mathbb{R}$$

$$p_k(v,U) = \text{number of cycles of length } k \text{ through vertex } v \text{ in } (U, E|U)$$

The subgraph $H = (C, E|C)$ induced by the set $C \subseteq V$ is a *p-core at level* $t \in \mathbb{R}$ iff $\forall v \in C : t \leq p(v, C)$ and C is a maximal such set.

Generalized cores algorithm

The function p is *monotone* iff it has the property

$$C_1 \subset C_2 \Rightarrow \forall v \in V : (p(v, C_1) \leq p(v, C_2))$$

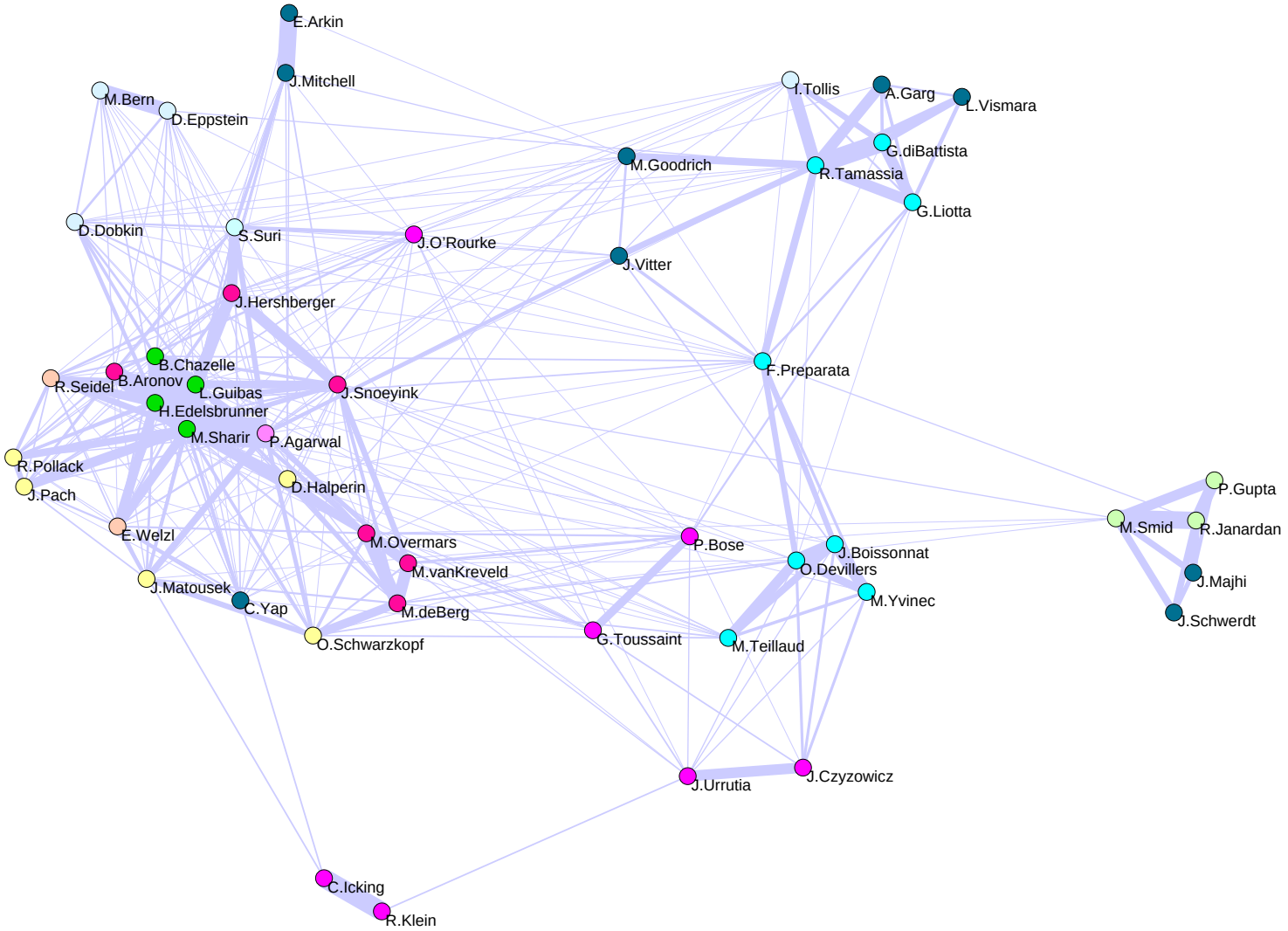
The degrees and the functions p_S, p_M and p_k are monotone. For a monotone function the p-core at level t can be determined, as in the ordinary case, by successively deleting vertices with value of p lower than t; and the cores on different levels are nested

$$t_1 < t_2 \Rightarrow H_{t_2} \subseteq H_{t_1}$$

The p-function is *local* iff $p(v, U) = p(v, \mathrm{adj}_U(v))$.

The degrees, p_S and p_M are local; but p_k is **not** local for $k \geq 4$. For a local p-function an $O(m \max(\Delta, \log n))$ algorithm for determining the p-core levels exists, assuming that $p(v, \mathrm{adj}_C(v))$ can be computed in $O(\deg_C(v))$.

p_S-core at level 46 of Geombib network



Triangular and short cycle connectivities (with M. Zaveršnik)

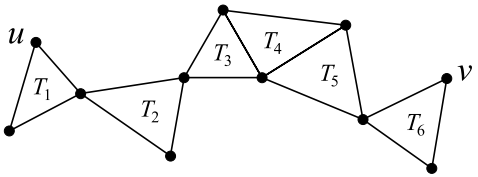
In this subsection we present an extension of notion of connectivity to connectivity by chains of triangles.

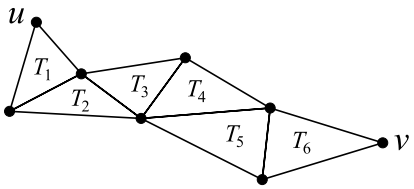
Undirected graphs

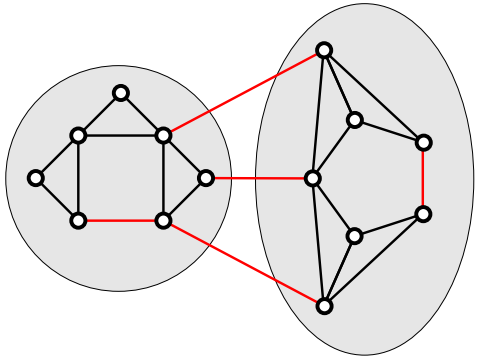
We call a *triangle* a subgraph isomorphic to K_3. A subgraph $H = (V', E')$ of $\mathbf{G} = (V, E)$ is *triangular* if each its vertex and each its edge belongs to at least one triangle in H.

A sequence $(T_1, T_2, \ldots, T_s)$ of triangles of $\mathbf{G}$ *(vertex) triangularly connects* vertices $u, v \in V$ iff $u \in T_1$ and $v \in T_s$ or $u \in T_s$ and $v \in T_1$ and $V(T_{i-1}) \cap V(T_i) \neq \emptyset$, $i = 2, \ldots s$. Such sequence is called a *triangular chain*. It *edge triangularly connects* vertices $u, v \in V$ iff a stronger version of the second condition holds $E(T_{i-1}) \cap E(T_i) \neq \emptyset$, $i = 2, \ldots s$.

Vertex and edge triangular connectivity



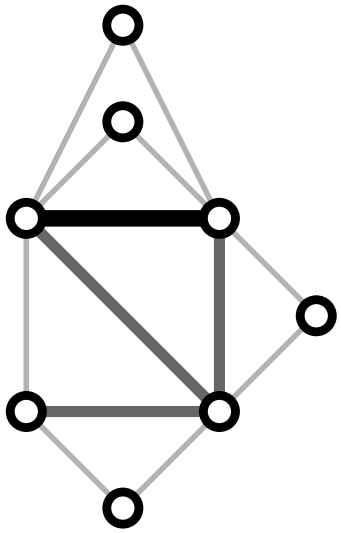




A pair of vertices $u, v \in V$ is *(vertex) triangularly connected* iff $u = v$, or there exists a chain that triangularly connects u and v. Triangular connectivity is an equivalence relation on the set of vertices V; and non-trivial triangular connectivity components are exactly maximal connected triangular subgraphs.

A pair of vertices $u, v \in V$ is *edge triangularly connected* iff $u = v$, or there exists a chain that edge triangularly connects u and v. Edge triangular connectivity components determine an equivalence relation on the set of edges E. Each nontriangular edge is in its own component.

Triangular network

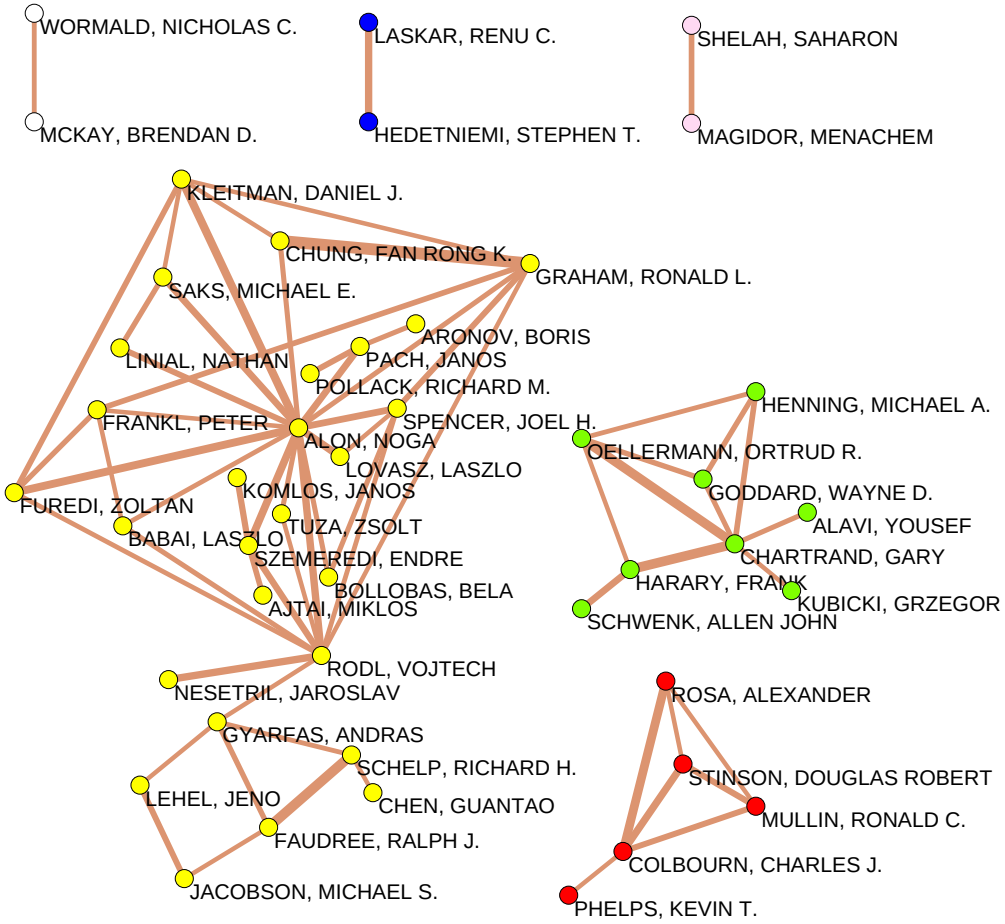


Let $\mathbf{G}$ be a simple undirected graph. A *triangular network* $\mathbf{N}_T(\mathbf{G}) = (V, E_T, w)$ determined by $\mathbf{G}$ is a subgraph $\mathbf{G}_T = (V, E_T)$ of $\mathbf{G}$ which set of edges E_T consists of all triangular edges of $E(\mathbf{G})$. For $e \in E_T$ the weight $w(e)$ equals to the number of different triangles in $\mathbf{G}$ to which e belongs.

A procedure for determining E_T and $w(e), e \in E_T$ simply collects all edges with $w(e) = |\mathrm{adj}(u) \cap \mathrm{adj}(v)| > 0$, $e = \{u, v\} \in E$. If the sets of neighbors $\mathrm{adj}(v)$ are ordered we can use merging to compute $w(e)$ faster. Nontrivial triangular connectivity components are exactly the components of $\mathbf{G}_T$.

Triangular networks can be used to efficiently identify dense clique-like parts of a graph. If an edge e belongs to a k-clique in $\mathbf{G}$ then $w(e) \geq k - 2$.

Edge-cut at level 16 of triangular network of Erdős collaboration graph



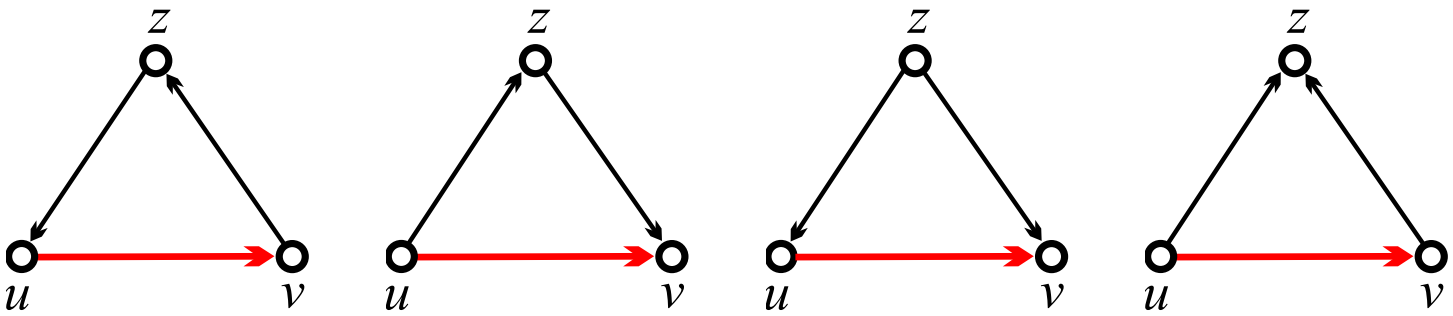
without Erdős,

$n = 6926,$

$m = 11343$

Directed graphs

If the graph $\mathbf{G}$ is mixed we replace edges with pairs of opposite arcs. In the following let $\mathbf{G} = (V, A)$ be a simple directed graph without loops. For a selected arc $(u, v) \in A$ there are four different types of directed triangles: **cyc**lic, **tra**nsitive, **in**put and **out**put.



cyc tra in out

For each type we get the corresponding triangular network $\mathbf{N}_{cyc}$, $\mathbf{N}_{tra}$, $\mathbf{N}_{in}$ and $\mathbf{N}_{out}$.

Cyclic triangular connectivity

A subgraph $H = (V', A')$ of $\mathbf{G}$ is *cyclic triangular* if each its vertex and each its arc belongs to at least one cyclic triangle in H. A connected cyclic triangular subgraph is also strongly connected.

A sequence $(T_1, T_2, \ldots, T_s)$ of cyclic triangles of $\mathbf{G}$ *(vertex) cyclic triangularly connects* vertex $u \in V$ to vertex $v \in V$ iff $u \in T_1$ and $v \in T_s$ or $u \in T_s$ and $v \in T_1$ and $V(T_{i-1}) \cap V(T_i) \neq \emptyset$, $i = 2, \ldots s$; such sequence is called a *cyclic triangular chain*. It *arc cyclic triangularly connects* vertex u to vertex v iff $A(T_{i-1}) \cap A(T_i) \neq \emptyset$, $i = 2, \ldots s$ holds; such sequence is called an *arc cyclic triangular chain*.

Again, we can introduce two types of cyclic triangular connectivity:

A pair of vertices $u, v \in V$ is *(vertex) cyclic triangularly connected* iff $u = v$, or there exists a cyclic triangular chain that connects u to v.

A pair of vertices $u, v \in V$ is *arc cyclic triangularly connected* iff $u = v$, or there exists an arc cyclic triangular chain that connects u to v.

. . . and transitive reachability

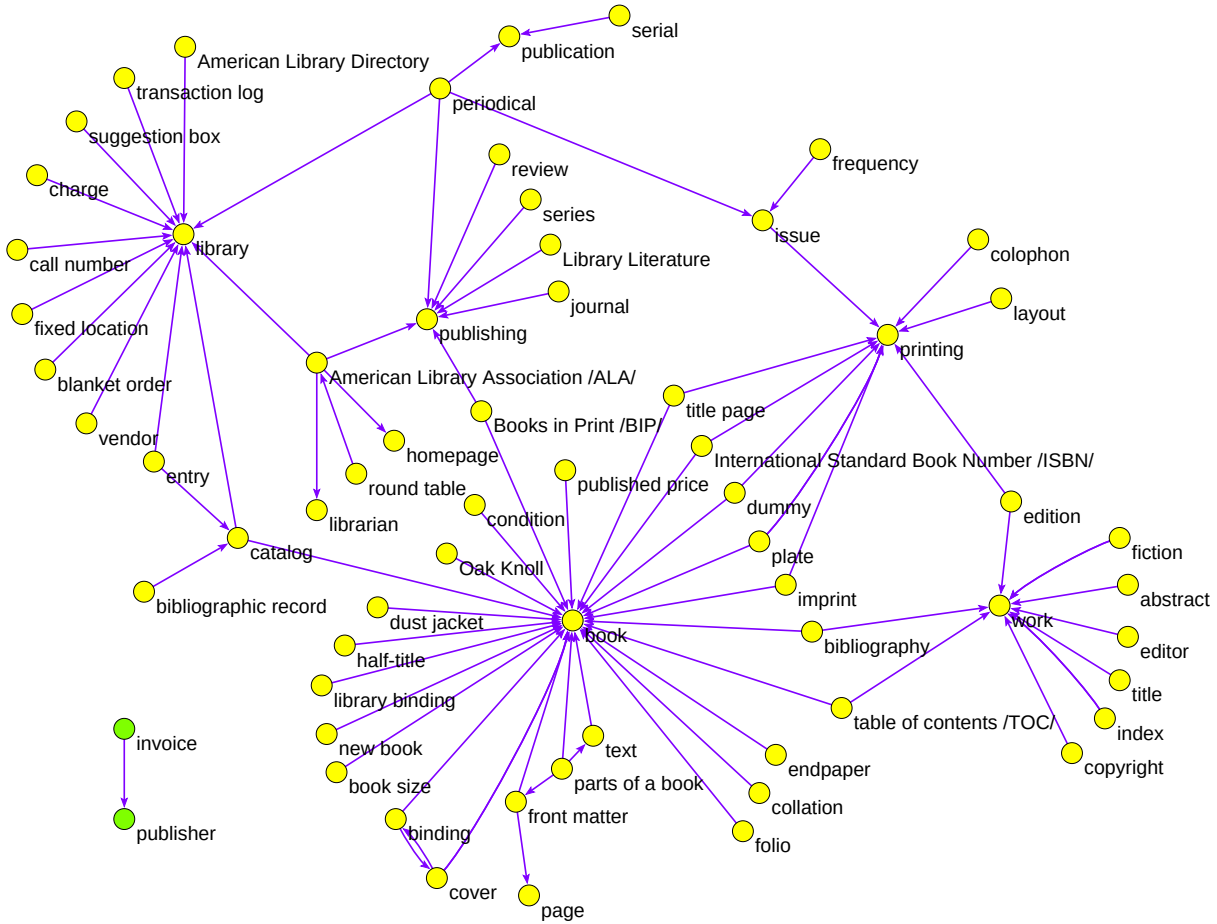
Cyclic triangular connectivity is an equivalence relation on the set of vertices V; and the arc cyclic triangular connectivity components determine an equivalence relation on the set of arcs A.

There exists also a parallel to unilateral connectivity. The vertex $v \in V$ is *transitively triangularly reachable* from the vertex $u \in V$ iff $u = v$, or there exists a walk from u to v in which each arc is transitive – is a base of some transitive triangle.

Transitive arcs are essentially reinforced arcs. If we remove from a graph $\mathbf{G} = (V, A)$ a transitive arc the reachability relation in V does not change.

These notions can be generalized to short cycle connectivity (see paper).

Edge-cut at level 11 of transitive network of ODLIS dictionary graph



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.

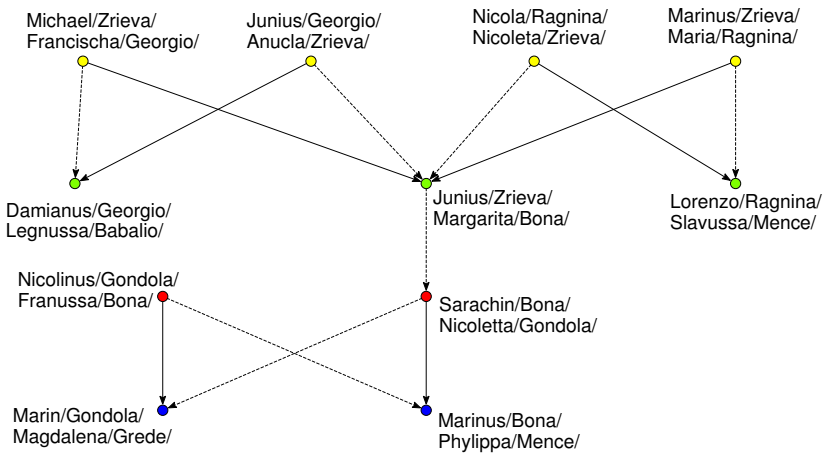
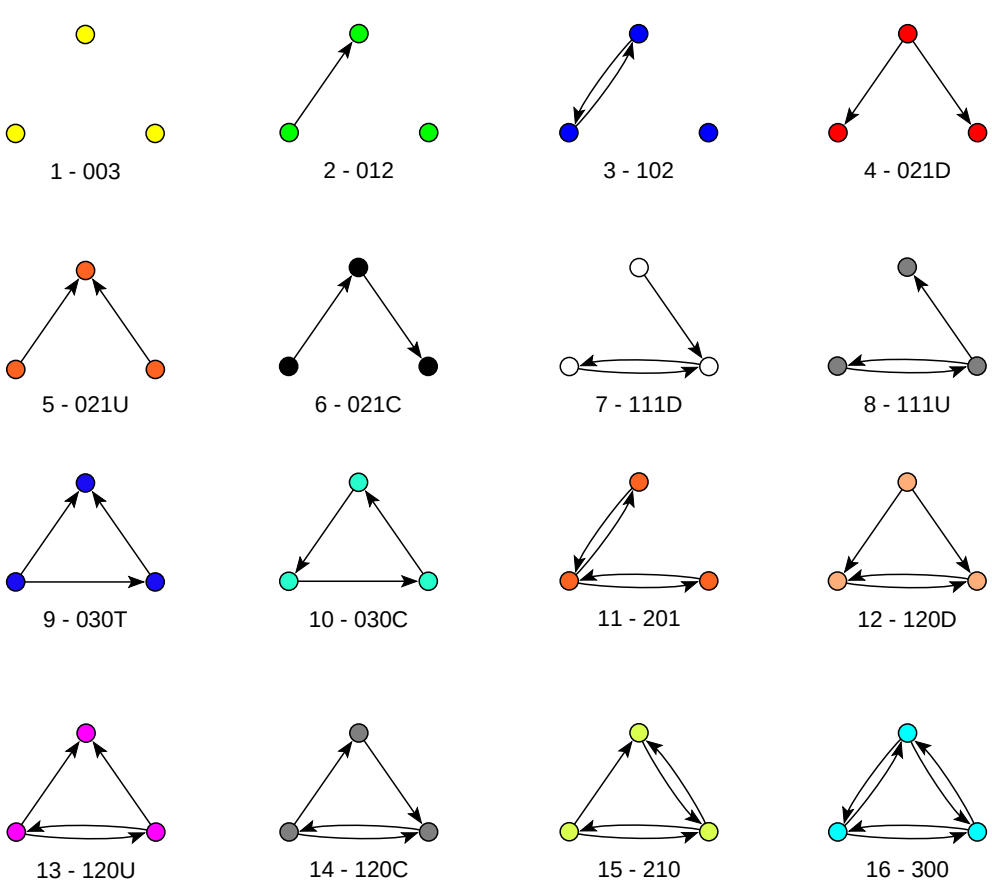


Figure presents three connected relinking marriages which are non-blood marriages found in the genealogy of ragusan noble families. The genealogy is represented as a p-graph. A solid arc indicates the _ *is a son of* _ relation, and a dotted arc indicates the _ *is a daughter of* _ relation. In all three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

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

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. The algorithm is based on the folllowing observation: *in a large and sparse graph most triads are null triads*. Let T_1, T_2, T_3 be the number of null, dyadic and connected triads. Since the total number of triads is $T = \binom{n}{3}$ and the above types partition the set of all triads, the idea of the algorithm is as follows:

- count all dyadic T_2 and all connected T_3 triads with their subtypes;
- compute the number of null triads $T_1 = T - T_2 - T_3$.

. . . Triadic spectrum

In the algorithm we have to assure that every non-null triad is counted exactly once while scanning the set of arcs. A set of three vertices $\{v, u, w\}$ can be in general selected in 6 different ways (v, u, w), (v, w, u), (u, v, w), (u, w, v), (w, v, u), (w, u, v). We solve the isomorphism problem by introducing the *canonical* selection that contributes to the triadic count; the other, noncanonical selections need not to be considered in the counting process.

The total complexity of the algorithm is $O(\hat{\Delta}m)$ and thus, for graphs with small maximum degree $\hat{\Delta} \ll n$, since $2m \leq n\hat{\Delta}$, of order $O(n)$.

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 $w : A \rightarrow \mathbb{R}$ 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 $A \subseteq U \times V$.

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 $\mathbf{A}^{(1)} = \mathbf{A}\mathbf{A}^T$, $a^{(1)}_{uv} = \sum_{z \in V} a_{uz} \cdot a^T_{zv}$. Evidently $a^{(1)}_{uv} = a^{(1)}_{vu}$. There is an edge $\{u, v\} \in E_1$ in $\mathbf{N}_1$ iff $N(u) \cap N(v) \neq \emptyset$. Its weight is $w_1(u, v) = a^{(1)}_{uv}$.

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

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

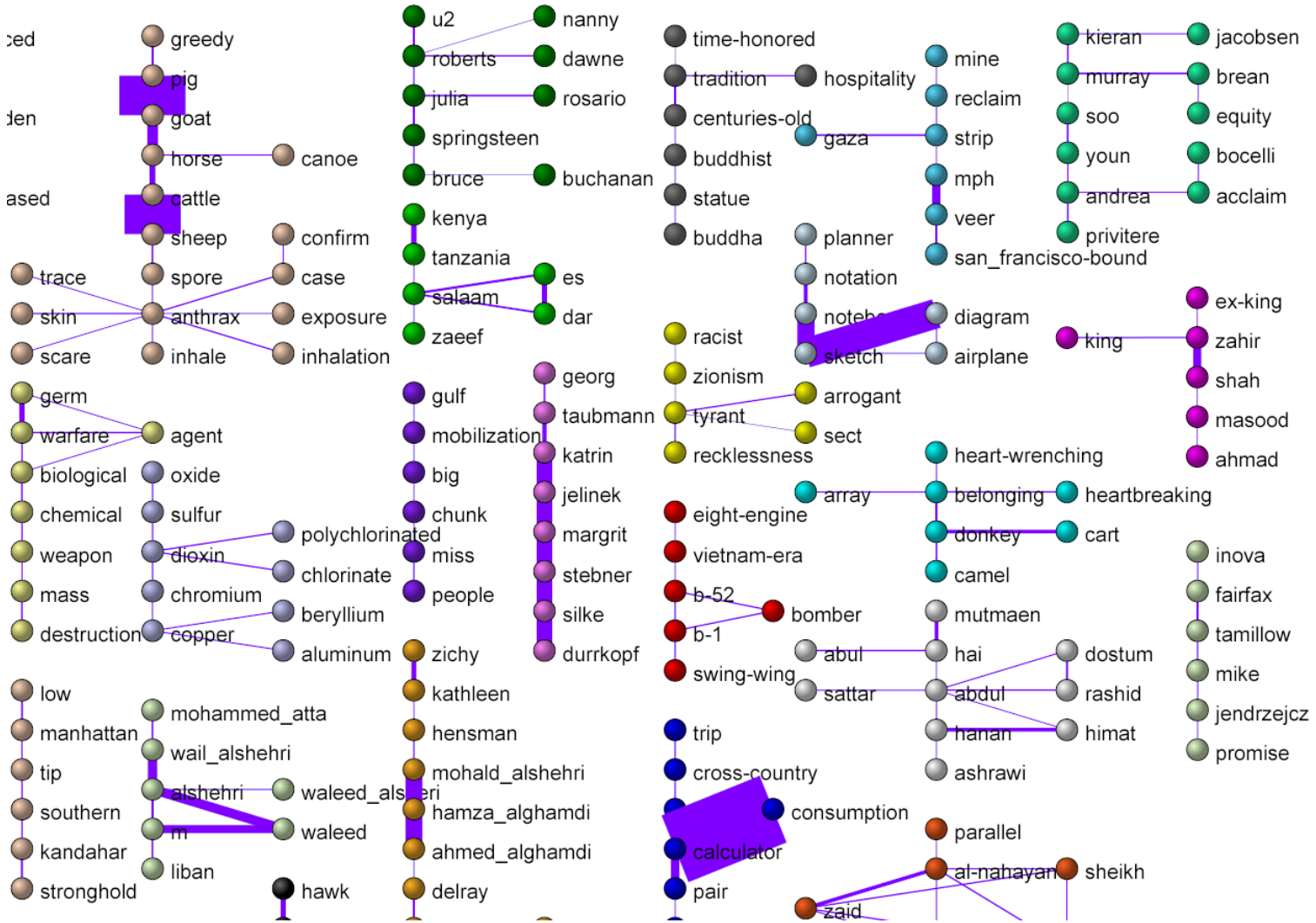
$$\text{Input}_{uv} = \frac{w_{uv}}{w_{vv}} \qquad \text{Output}_{uv} = \frac{w_{uv}}{w_{uu}}$$

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

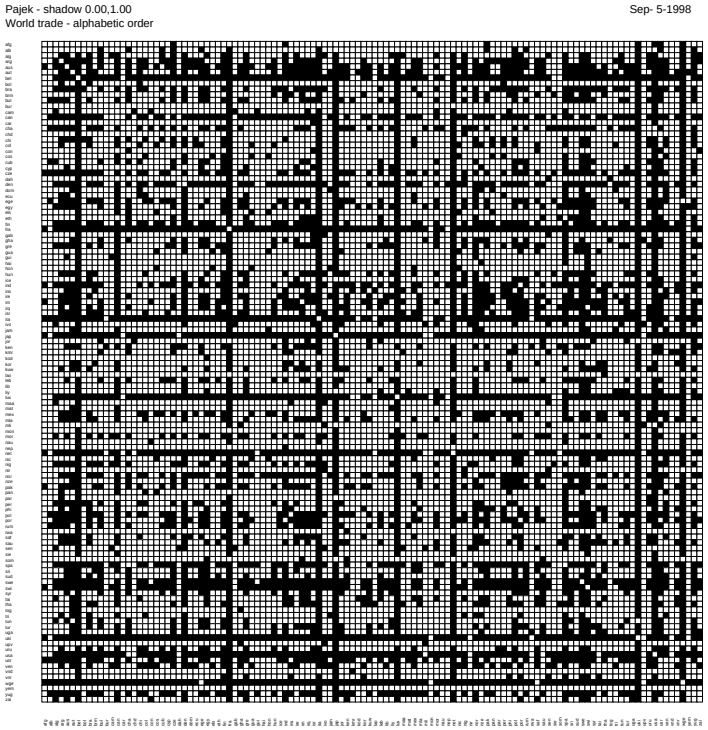
$$\text{MinDir}_{uv} = \begin{cases} \frac{w_{uv}}{w_{uu}} & w_{uu} \leq w_{vv} \\ 0 & \text{otherwise} \end{cases} \qquad \text{MaxDir}_{uv} = \begin{cases} \frac{w_{uv}}{w_{vv}} & w_{uu} \leq w_{vv} \\ 0 & \text{otherwise} \end{cases}$$

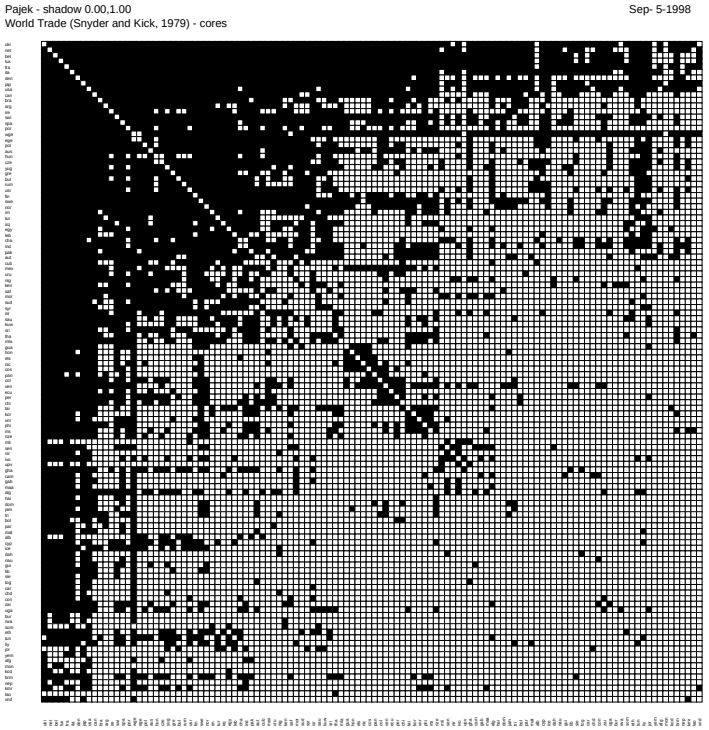
After a selected normalization the important parts of network are obtained by edge-cutting the normalized network at selected level t and preserving components with at least k vertices.

GeoDeg normalization of Reuters terror news network



Generalized blockmodeling





In the figure the *Snyder and Kick's world trade network* is presented by its matrix: on the left side the units (states) are ordered in the alphabetic order of their names; on the right side they are ordered on the basis of clustering results. It is evident that a 'proper' ordering can reveal a structure in the network.

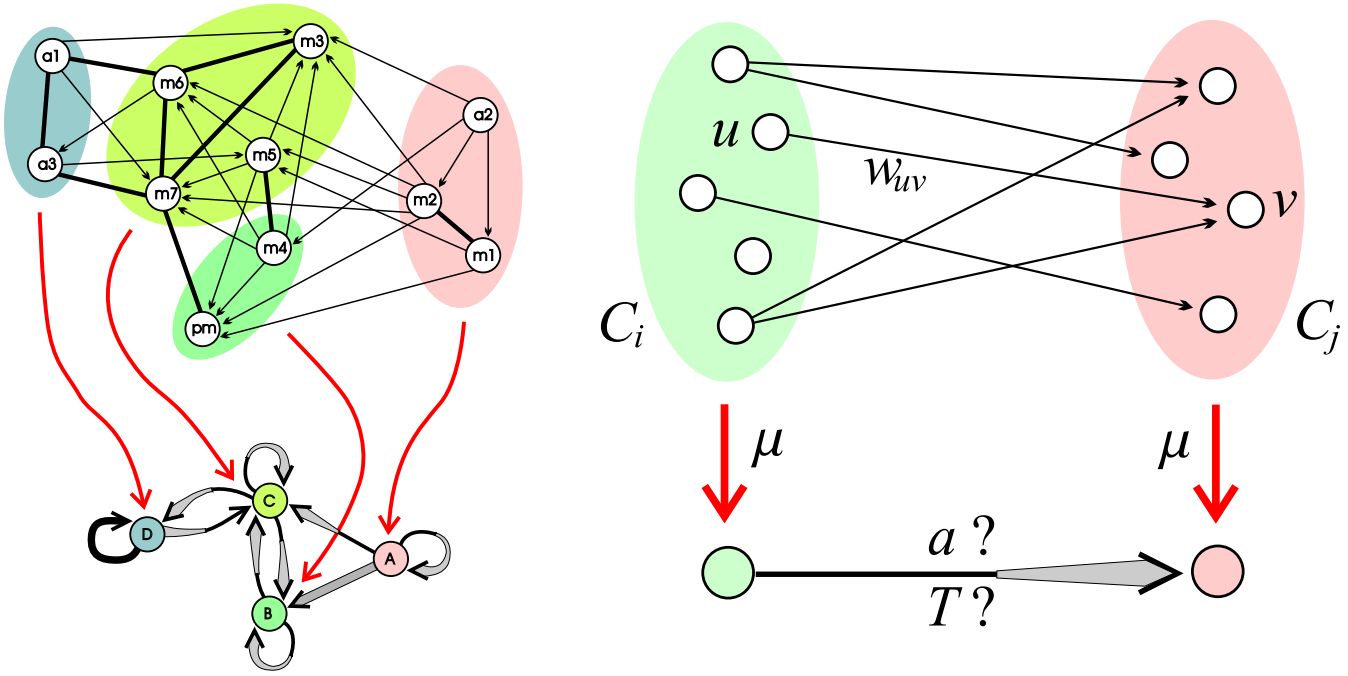
Clusters and blocks

On the networks of moderate size (up to some hundreds of units) such orderings can be produced also using the *blockmodeling* methods. Their goal is to reduce a large, potentially incoherent network to a smaller comprehensible structure that can be interpreted more readily.

One of the main procedural goals of blockmodeling is to identify, in a given network $\mathbf{N} = (\mathbf{U}, R)$, $R \subseteq \mathbf{U} \times \mathbf{U}$, *clusters* (classes) of units/ vertices that share structural characteristics defined in terms of R. The units within a cluster have the same or similar connection patterns to other units. They form a *clustering* $\mathbf{C} = \{C_1, C_2, \ldots, C_k\}$ which is a *partition* of the set $\mathbf{U}$. Each partition determines an equivalence relation (and vice versa).

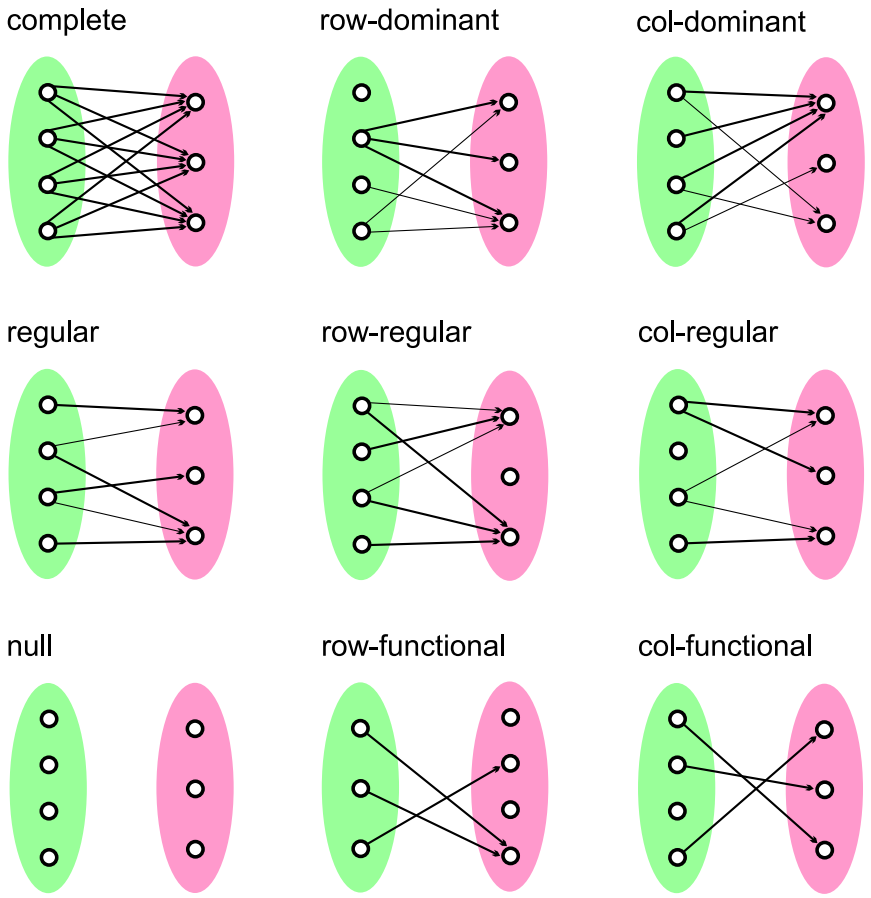
A clustering $\mathbf{C}$ partitions also the relation R into *block*s $R(C_i, C_j) = R \cap C_i \times C_j$. Each such block consists of units belonging to clusters C_i and C_j and all arcs leading from cluster C_i to cluster C_j. If $i = j$, a block $R(C_i, C_i)$ is called a *diagonal* block.

Blockmodel



A *blockmodel* consists of structures obtained by identifying all units from the same cluster of the clustering **C**. For an exact definition of a blockmodel we have to be precise also about which blocks produce an arc in the *reduced graph* and which do not, and of what *type*.

Block Types



In the figure some types of connections of blocks are presented. The reduced graph can be represented by relational matrix, called also *image matrix*.

Also, by *reordering* of network matrix so that the units from each cluster of the optimal clustering are located together we obtain a matrix representation of the network with visible structure.

Blockmodeling as a clustering problem

How to determine an appropriate blockmodel? The blockmodeling can be formulated as a *clustering problem* (Φ, P) as follows:

Determine the clustering $\mathbf{C}^\star \in \Phi$ for which

$$P(\mathbf{C}^\star) = \min_{\mathbf{C} \in \Phi} P(\mathbf{C})$$

Since the set of units $\mathbf{U}$ is finite, the set of feasible clusterings Φ is also finite. Therefore the set $\mathrm{Min}(\Phi, P)$ of all solutions of the problem (optimal clusterings) is not empty. In theory, the set $\mathrm{Min}(\Phi, P)$ can be determined by the complete search – but it turns out that most cases of the clustering problem are $\mathcal{NP}$ hard. The blockmodeling problems are usually solved using local optimization methods based on moving a unit from one cluster to another or interchanging two units between two clusters.

Blockmodeling criterion function

One of the possible ways of constructing a criterion function that directly reflects the considered equivalence is to measure the fit of a clustering to an ideal one with perfect relations within each cluster and between clusters according to the considered equivalence.

Given a clustering $\mathbf{C} = \{C_1, C_2, \ldots, C_k\}$, let $\mathcal{B}(C_u, C_v)$ denote the set of all ideal blocks corresponding to block $R(C_u, C_v)$. Then the global error of clustering $\mathbf{C}$ can be expressed as

$$P(\mathbf{C}) = \sum_{C_u, C_v \in \mathbf{C}} \min_{B \in \mathcal{B}(C_u, C_v)} d(R(C_u, C_v), B)$$

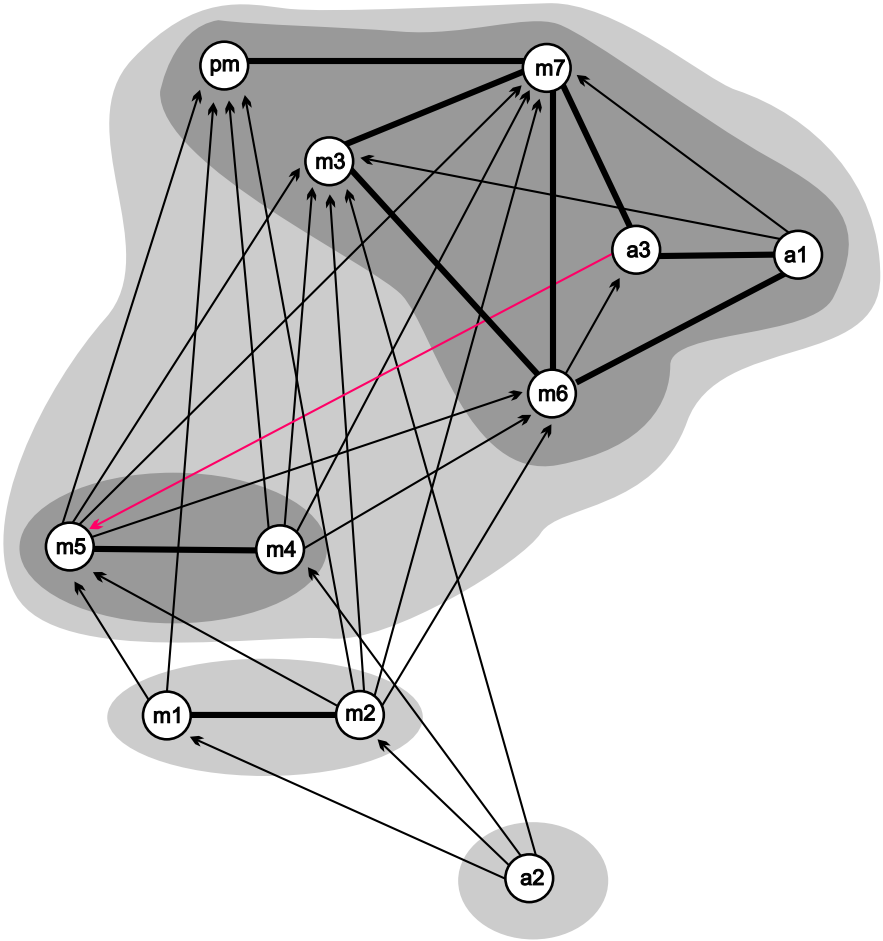
where the term $d(R(C_u, C_v), B)$ measures the difference (error) between the block $R(C_u, C_v)$ and the ideal block B. d is constructed on the basis of characterizations of types of blocks. The function d has to be compatible with the selected type of equivalence. Determining the block error, we also determine the type of the best fitting ideal block (the types are ordered).

. . . criterion function

The criterion function $P(\mathbf{C})$ is *sensitive* iff $P(\mathbf{C}) = 0 \Leftrightarrow \mathbf{C}$ determines an exact blockmodeling. For all presented block types sensitive criterion functions can be constructed.

Once a clustering $\mathbf{C}$ and types of blocks are determined, we can also compute the values of connections by using averaging rules.

A Symmetric Acyclic Blockmodel of Student Government



In the figure a symmetric acyclic (edge connected inside clusters, acyclic reduced graph) blockmodel of Student Government at the University of Ljubljana is presented.
The obtained clustering in 4 clusters is almost exact. The only error is produced by the arc $(a3, m5)$.

In preparation

Generating large random networks

Links

Web version of these slides

http://vlado.fmf.uni-lj.si/pub/networks/doc/seminar/paris03.pdf

Pajek home page

http://vlado.fmf.uni-lj.si/pub/networks/pajek/

Papers and slides about **Pajek**

http://vlado.fmf.uni-lj.si/pub/networks/doc/

Pajek data sets

http://vlado.fmf.uni-lj.si/pub/networks/data/