Pajek

Pajek is a program package for Windows 32 and 64, which enables analyses of *large networks*. Program is freely available at:
http://mrvar.fdv.uni-lj.si/pajek

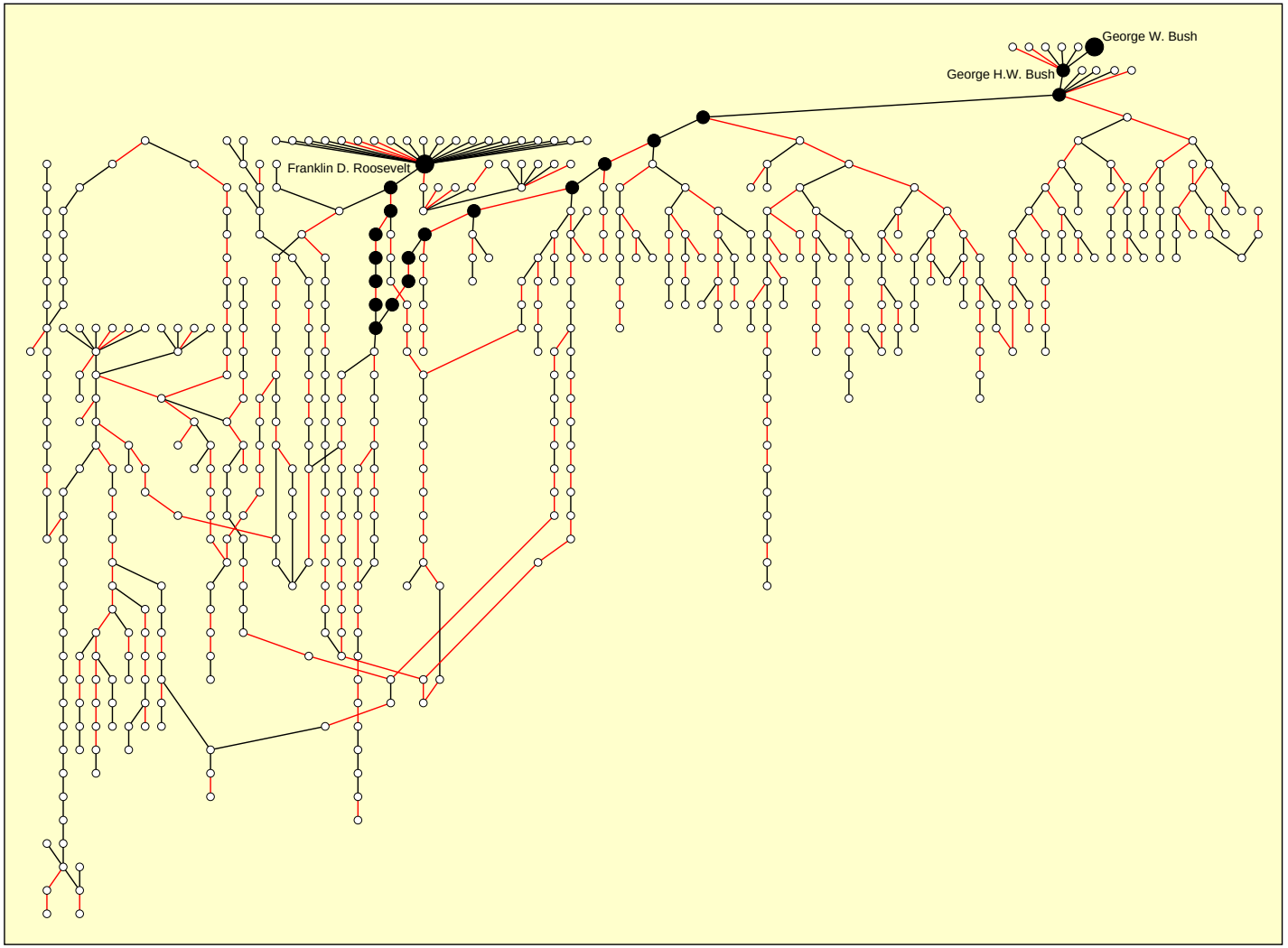
Analyses in Pajek are performed using six data structures:

1. network – main object (vertices and lines):
   graph, valued network, 2-mode or temporal network
2. partition – nominal property of vertices (gender);
3. vector – numerical property of vertices;
4. permutation – reordering of vertices;
5. cluster – subset of vertices (e.g. a cluster from partition);
6. hierarchy – hierarchically ordered clusters and vertices.

Examples

- flor.net,
- 1crn.net,
- football.net,
- davis.net (two mode network)

The largest connected component in the genealogy of American presidents

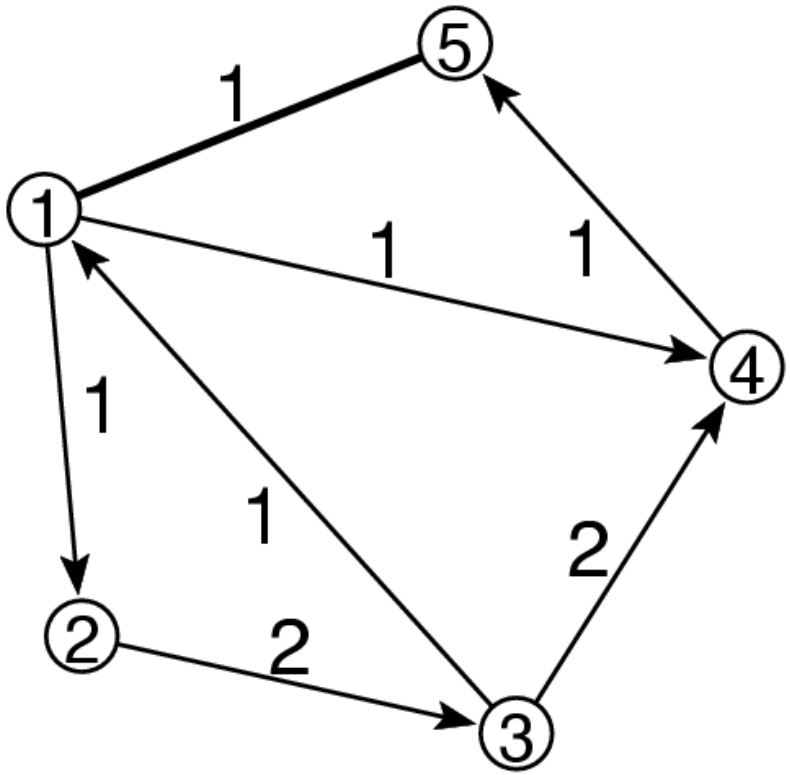


Network – .NET

Network can be defined in different ways on input file. Look at three of them:

1. List of neighbors (Arcslist / Edgeslist)

```
*Vertices 5
1 ”a”
2 ”b”
3 ”c”
4 ”d”
5 ”e”
*Arcslist
1 2 4
2 3
3 1 4
4 5
*Edgeslist
1 5
```



Explanation: Data must be prepared in an input (ASCII) file. Program *NotePad* can be used for editing. Much better is a shareware editor, *TextPad* (http://www.textpad.com/).

Words, starting with *, must always be written in first column of the line. They indicate the start of a definition of vertices or lines.

Using *Vertices 5 we define a network with 5 vertices. This must always be the first statement in definition of a network.

Definition of vertices follows after that – to each vertex we give a label, which is displayed between ” and ”. If labels of vertices are equal to their sequential numbers this part can be omitted.

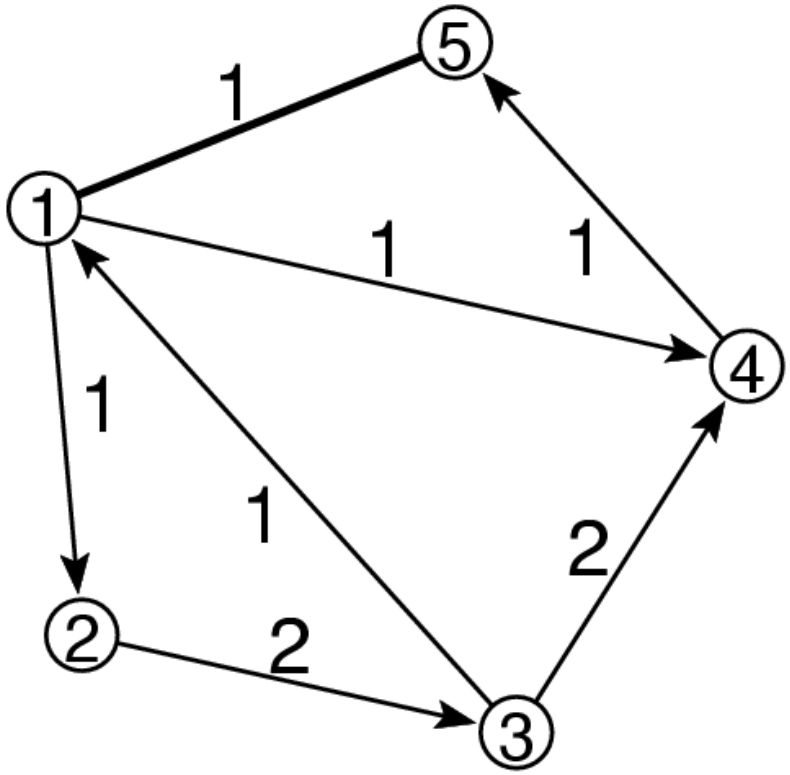
Using *Arcslist, a list of directed lines from selected vertices are declared (1 2 4 means, that there exist two lines from vertex 1, one to vertex 2 and another to vertex 4).

Similarly *Edgeslist, declares list of undirected lines from selected vertex.

In the file no empty lines are allowed – empty line means end of network.

2. Pairs of lines (Arcs / Edges)

*Vertices 5
1 ”a”
2 ”b”
3 ”c”
4 ”d”
5 ”e”
*Arcs
1 2 1
1 4 1
2 3 2
3 1 1
3 4 2
4 5 1
*Edges
1 5 1



Explanation:

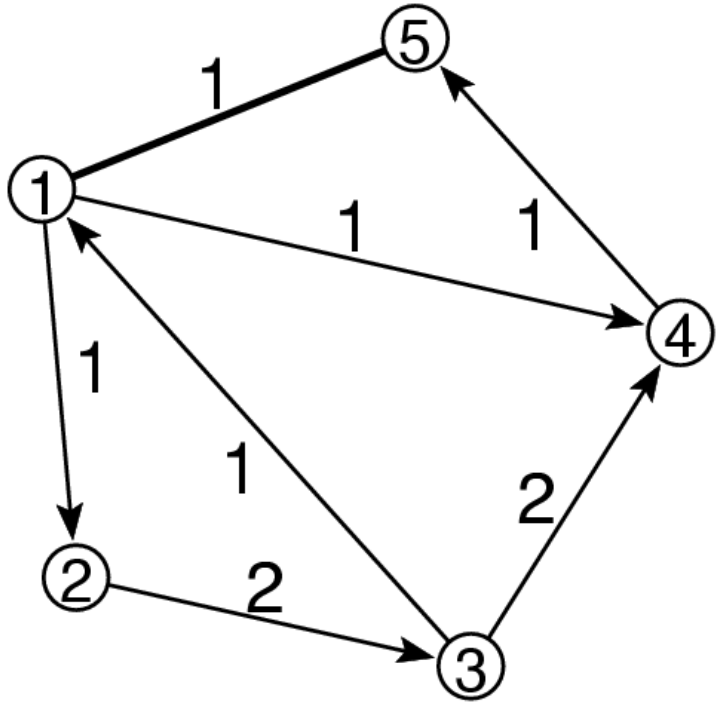
This is the most general format. In this case every arc/edge is defined separately in new line - initial and terminal vertex of every arc/edge are given. Directed lines are defined using *Arcs, undirected lines are defined using *Edges. The third number in rows defining arcs/edges gives the value of the arc/edge. Arcs from 2 to 3 and from 3 to 4 have value 2, all others have value 1.

In the previous format (Arcslist / Edgeslist) values of lines cannot be defined – the format is suitable only if all values of lines are 1.

If values of lines are not important the third number can be omitted (all lines get value 1).

3. Matrix

```
*Vertices 5
1 ”a”
2 ”b”
3 ”c”
4 ”d”
5 ”e”
*Matrix
0 1 0 1 1
0 0 2 0 0
1 0 0 2 0
0 0 0 0 1
1 0 0 0 0
```



Explanation:

In this format directed lines (arcs) are given in the matrix form (*Matrix). If we want to transform bidirected arcs to edges we can use Network/Create New Network/Transform/Arcs to Edges/Bidirected only

Only those elements necessary to define structure of network were described so far. Additionally, Pajek enables precise definition of elements used for drawing networks (coordinates of vertices in space, shapes and colors of vertices and lines, ...). E.g.: The shape of vertices in Pajek is determined in the input file, so that after the vertex label (and coordinates) we add: box, ellipse, diamond, triangle or empty:

```
*Vertices 5
1 "a" box
2 "b" ellipse
3 "x" diamond
4 "y" triangle
...
```

Among internal formats (.NET), Pajek recognizes several other formats: UCINET DL; Vega; GEDCOM and some chemical formats: BS (Ball and Stick), MAC (Mac Molecule) and MOL (MDL MOLfile).

Interactive definition of networks

Simple networks can be defined inside program Pajek as well without definition in an input file:

First the empty network (network without lines) is built, later lines are added.

Select: Network/Create New Network/Empty Network
In our example with 5 vertices we input:
Enter number of vertices: 5

Then we select Draw/Network (or press appropriate icon or Ctrl+G). The network is represented by layout in a new window. Vertices can be moved to other positions by clicking with left mouse button on selected vertex, holding the mouse button down and moving the mouse.

After that we define for every vertex all lines connected to the vertex: We click with right mouse button on selected vertex. A new window is opened, in which a double click on Newline vertices neighboring a selected vertex

are entered.

If an arc is going from a selected vertex to vertex 2, we input -2, if an arc is coming from vertex 2, we enter $+2$, if the two vertices are connected by an edge we enter only 2.

The value of a line is entered by clicking on the highlighted selection with the right mouse button in the same window and entering the desired value.

A line can be deleted by double clicking the highlighted selection in the same window with the left mouse button.

The Draw window can be refreshed using the command Redraw.

Some commands in Draw window

All commands are described in the html file (Manual) that is available on the Pajek homepage.

- **Options/Mark vertices using** – selects the way vertices are marked in the picture
- **Options/Lines** – visibility or nonvisibiliy of arcs and edges, selects the way lines are marked in the picture
- **Options/Size** – selects the size of vertices, size of font, size of arrows and width of lines
- **Options/Colors** – selects background color, color of vertices, lines, font. . .

Determining layouts of networks

The imagination of a network is often obtained only by a picture of it. Several automatic and manual drawing routines are included in Pajek.

Automatic layout generation

- **Energy** – Idea: the network is represented like a physical system, and we are searching for the state with minimal energy. Two algorithms:
  - **Layout/Energy/Kamada-Kawai** – slower, but using **Separate Components**, you can tile connected components in plane.
  - **Layout/Energy/Fruchterman-Reingold** – faster, drawing in a plane or space (2D or 3D), and selecting the repulsion factor
- **Pivot MDS, VOS Mapping** – Additional 2 or 3 dimensional drawings. Pivot MDS works fine also for networks with up to 100.000 vertices, VOS Mapping for denser networks. Nice pictures are usually obtained if there are symmetries in the network.

Manual drawing

We can move the vertex by clicking with left mouse button on a selected vertex and moving the mouse.

Pictures in space: a picture can be rotated by pressing (holding down) keys x, X, y, Y, z, Z (the key stands for axis of rotation, small/capital letters mean positive or negative orientation).

Any axis of rotation can be selected using **Spin/Normal**. After that, rotation around the selected axis is executed by holding down s or S.

A part of the network can be zoomed by selecting a window with the picture using the right mouse button. The complete picture is obtained using Redraw.

If we want to save changes for a network (in our case the picture), we select the icon for saving the network or **File/Network/Save**, and then choose the type of presentation and the name of the file. Types of presentation were defined in the beginning (input formats). If the name is equal to the name

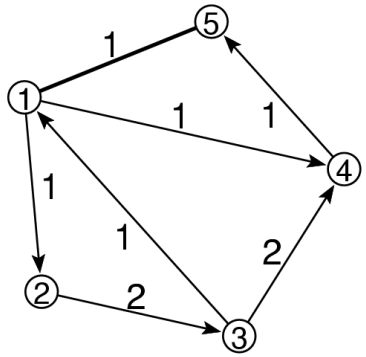
of the network that already exists, the old network is lost.

If we select Arc/Edges presentation, the new file will look like (the difference is only that coordinates of vertices are added):

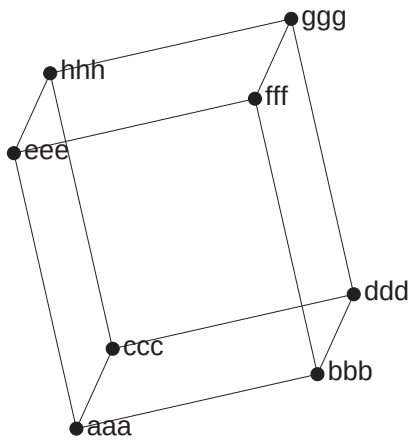
```
*Vertices 5
1 ”a”     0.1672   0.3272   0.5000
2 ”b”     0.2029   0.7394   0.5000
3 ”c”     0.6144   0.8334   0.5000
4 ”d”     0.8328   0.4794   0.5000
5 ”e”     0.5565   0.1666   0.5000
*Arcs
1 2 1
1 4 1
2 3 2
3 1 1
3 4 2
4 5 1
*Edges
1 5 1
```

Examples

1. Build the following network interactively in Pajek:



2. Prepare the input file ex1.net with the description of network in the shape of cube, with the labels of vertices:
   aaa, bbb, ccc, ddd, eee, fff, ggg and hhh: Read the network, draw it in plane and space, and save it again.



3. Read the network that is stored in the file ex2.net and try all algorithms for automatic layout generation.

4. class1.net: A network consists of 15 undergraduate students (box-boys, ellipse-girls) attending lectures on Social network analysis at Faculty of Social Sciences, University of Ljubljana (2002). Students were asked: *from whom would you borrow learning materials*. The number of choices was not limited. To get partition by shapes of vertices use
   Network/Create Partition/Vertex Shapes

Partitions

Partitions are used to describe nominal properties of vertices (e.g., 1-men, 2-women). The default extension for a partition is .CLU – clustering.

If we have a network with 5 vertices, and want to place vertices 1 and 5 into cluster 1, and all other vertices in cluster 2, the corresponding partition is described on input file in the following way:

```
*Vertices 5
1
2
2
2
1
```

We can build a partition using Pajek too, so that we first use
Partition/Create Constant Partition 0
The dimension of partition is by default set to the number of vertices in the selected network. In this way we put all vertices in cluster 0. Then we select File/Partition/Edit/ or press the corresponding icon, and for each unit input its cluster number.

If we use Draw/Network+First Partition or press Ctrl+P colors of vertices will be used to show to which cluster each vertex belongs.

The second possibility to determine a partition is:

- Select Draw/Network+Create Null Partition or press Ctrl+A. Using that command three operations are executed: a new partition of equal dimension as the number of vertices is generated; all vertices are put to cluster 0 and the network is drawn using the obtained null partition (all vertices are cyan).

- In the picture of a network: By pressing the middle mouse button we increment the cluster number of selected vertex (if we press the Alt key on the keyboard at the same time, we decrement the cluster number).

  If the mouse has only two buttons, the left mouse button and Shift key are used to increment, and the left mouse button and Alt key are used to decrement the cluster number.

  If only the part of network is selected we increment/decrement the cluster number of all selected vertices by pressing with mouse somewhere in the picture (not on a vertex).

Colors used are shown in
Options/Colors/Partition Colors/for Vertices in Draw window.

In the layout of a network, all vertices that belong to a selected cluster can be moved at once by pressing the left mouse button close to the vertex of selected color, holding the mouse down and moving the mouse.

Example: usair.net + usair.clu, import.net + cont.clu.

Vectors

Vectors are used to describe numerical properties of vertices (e.g., centralities). The default extension for a vector is .VEC. Example:

```
*Vertices 5
0.75
0.12
0.1234
1.234
0.03
```

If we use Draw/Network+First Vector sizes of vertices will be proportional to vector values. If we use Draw/Network+First Vector+SecondVector x-size will be determined by the first and y-size by the second vector. For the later two Vectors must be first made visible in the main Pajek window by pressing **Vectors** button. Vertices of different colors and sizes may be obtained using Draw/Network+Partition+ First Vector or .../+Second Vector

Example: import.net + cont.clu + GDP1995.vec.

Pajek project files

Often it is the case that not only network but also several properties of the vertices are known in advance. This properties are usually stored as partitions or vectors. It is time consuming to load objects one by one. Therefore it is convenient to store all data in one file, called Pajek project file (.PAJ). For example store selected network and several properties saved as partitions and vectors in one file.

Project files can be produced manually by pasting several data objects in one file or inside Pajek using **File/Pajek Project File/Save** – all currently loaded objects will be saved in selected project file. It is recommended to rename captions of objects.

To load objects stored in Pajek project file select **File/Pajek Project File/Read**

Example: world_trade.paj.

Output formats

The obtained pictures in a plane or space can be stored in output formats that are suitable for including pictures in text processors (e. g. Word) or examining using special viewers. Saving in output formats is available in option **Export**. The possible formats are:

2D exports

- EPS – The picture can be shown using program GSView or included in text editors, e.g. LaTeXor Word (example write.net).
- SVG – *Scalable Vector Graphics* – it is vector graphics format (like EPS) – the picture can be resized without loosing the quality. SVG is recognized by Internet browsers like Explorer using special plug-in. A free copy of the plugin for Explorer can be downloaded from:
  http://www.adobe.com/svg/viewer/install/
  In FireFox SVG you do not need a plugin. The picture in SVG can be further edited using InkScape (http://inkscape.org/).

Several useful improvements of the picture can be done using this editor, like drawing contours and shading area with transparent colors around selected vertices, adding legends and other labels. Inkscape can also be used for transforming the obtained SVG picture to other formats, like EPS and PDF.

example write.net.

- JPEG (JPG) – Export to JPG in higher or lower quality and in colors on greyscale.
- Bitmap (BMP) – It is just a snapshot of the picture and can be quite large in size (example write.net).

3D exports

- X3D – for pictures in space. X3D is an XML based 3D computer graphics. The main advantage of the representation is that the layout is not statical but dynamical – we can travel in the picture, rotate the picture, select different views...

  Plug-ins available InstantPlayer, FluxPlayer, SwirlX3D...
  Example ex2.net.

- Kinemages (*KINEmatic iMAGES*)
  The program Mage is freely available at:
  http://www.prosci.org/Kinemage/MageSoftware.html
  it is more powerful than Chime – simple animation can be done too (example ex2.net).

- VRML - Virtual Reality Modeling Language – for pictures in space. VRML is not supported anymore, X3D is the enhanced successor to VRML. VRML is recognized by Internet browsers like Explorer using

special plug-in CosmoPlayer. Cortona can be used as well:
http://www.parallelgraphics.com/products/cortona/
Example ex2.net.

- MDL MOLfile (Chime viewer)
  http://www.mdli.com/download/chimedown.html
  The format is primarily intended for space pictures of molecules, but it can also be used for drawing undirected networks too (example ex2.net).

Examples: ex2, ex4...

Nice pictures of networks

Some properties of nice pictures of networks:

- not too many crossings of lines,
  a graph that can be drawn without crossing of lines is called *a planar* graph)
- not too many small angles among lines that have one vertex in common
- not too long or too short lines (all lines approximately of the same length)
- vertices should not be too close to lines

In Pajek coordinates of vertices are numbers between 0 and 1. The decimal delimiter is a period. In the case that coordinates are from some other interval, the picture can be extended/shrunk to the size of the window using Options/Transform/Fit Area.

Basic information about a network

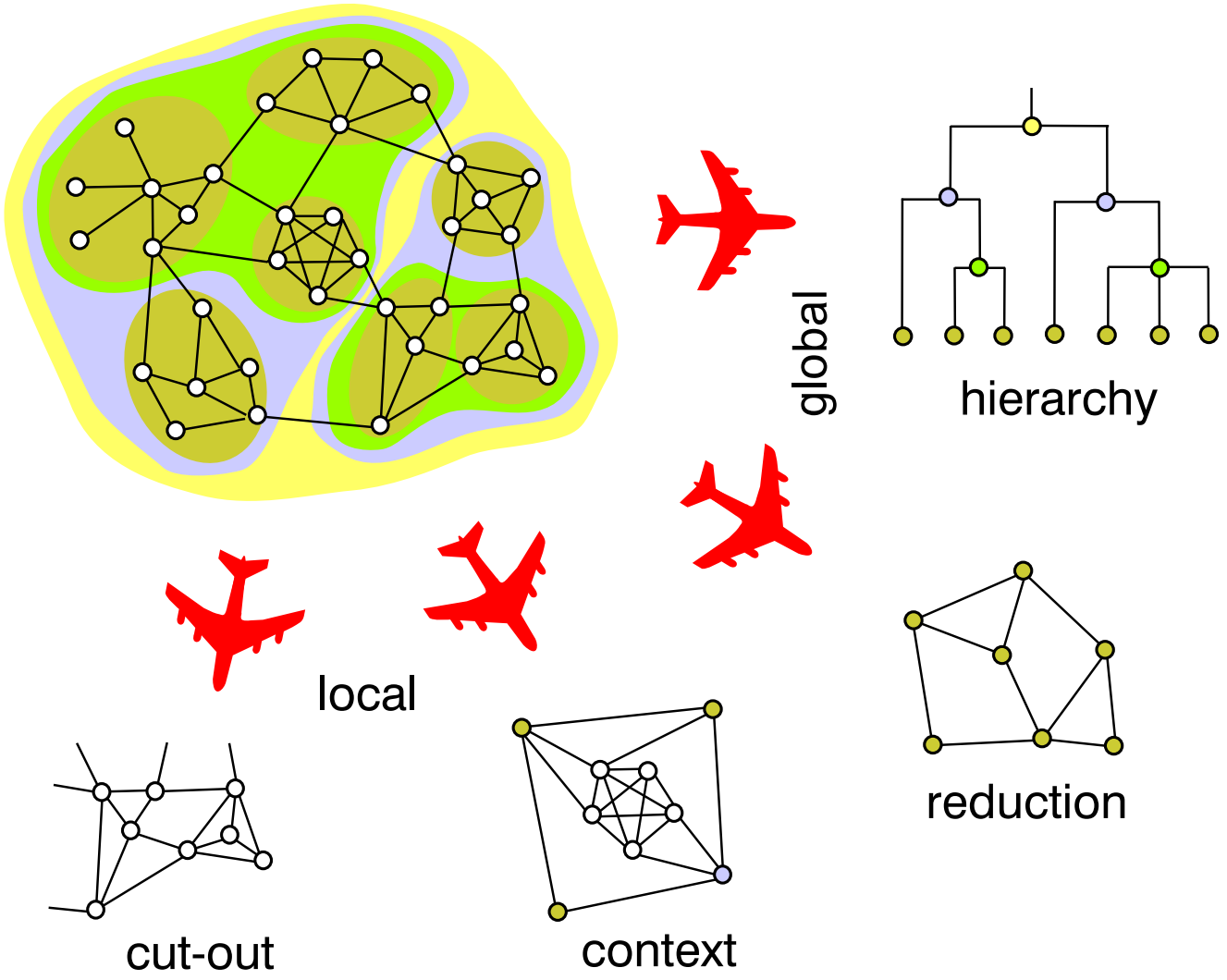
Basic information about a network can be obtained by Network/Info/General which is available in the main window of the program. We get

- number of vertices
- number of arcs, number of directed loops
- number of edges, number of undirected loops
- density of lines

Additionally we must answer the question:
Input 1 or 2 numbers: +/highest, -/lowest
where we enter the number of lines with the highest/lowest value or interval of values that we want to output.

If we enter $\boxed{10}$, 10 lines with the highest value will be displayed. If we enter $\boxed{-10}$, 10 lines with the lowest value will be displayed. If we enter $\boxed{3\ \ 10}$, lines with the highest values from rank 3 to 10 will be displayed.

Local and global views on network



Example: shr1.net, shr.clu

Local view is obtained by extracting subnetwork induced by selected cluster of vertices.
Students in the class: relations among boys (girls) only.

Global view is obtained by shrinking vertices in the same cluster to new (compound) vertex. In this way relations among clusters of vertices are shown.
Students in the class: compound relation between boys and girls (number of arcs between the two groups).

Combination of local and global view is **contextual view**: Relations among clusters of vertices and selected vertices are shown.
Students in the class: for every girl – what is the number of arcs pointing to boys.

Example: Import and export among countries

Import and export in 1994 among 80 countries are given. They is given in 1000$, but imports from countries which are lower than 1% of the total import of a country are not considered. File: import.net.
Partition according to continents – cont.clu: 1 – Africa, 2 – Asia,
3 – Europe, 4 – N. America, 5 – Oceania, 6 – S. America.

Extracting subnetworks

We are also interested in the subnetwork (vertices and lines) which the vertices in a selected cluster induce. In this case we use:
Operations/Network+Partition/Extract SubNetwork

Before we run this operation we must select the network and partition in the main window. Additionally we must give the interval of clusters which we want to extract.

Example: Import and export inside selected continent.

Reduction – shrinking

Example: import/export on the level of continents.

Operations/Network+Partition/Shrink Network. Pajek asks for minimum number of lines, that must exist among shrunk vertices that will generate a line among shrunk vertices. We can use the default value (1). After that we are asked, which cluster we do not want to shrink. If we want to shrink all clusters, we input number of cluster that does not exist (in our case for example 0, or 7, or . . .).

Similarly the ***context*** is obtained: we select the cluster number that we do not want to shrink.

Removing lines with low values

Another way to reduce size of networks is according to line values. Take for example import.net. Using Network/Info/Line Values and typing number of classes (e.g. #10) gives us frequency distribution of line values.

We can remove all lines with values lower than the specified one by Network/Create New Network/Transform/Remove/Lines with value/lower than and typing the threshold value, e.g. 340000. Look at the picture of the obtained network.

Example of a large network: connections among words in dictionary

A large network can be generated from words of dictionary. Two words are connected using an undirected line if we can reach one from the other by

- changing a single character (e. g., work – word)
- adding / removing a single character (e. g., ever – fever).

Knuth's dictionary was used. There exist 52,652 words having 2 to 8 characters. The obtained network has 92,307 edges. The network is sparse: density is 0.0000665.

File: dic28.net.

Shortest paths

We can find **one shortest path** between two vertices using
Network/Create New Network/SubNetwork with Paths/...
...One Shortest Path between Two Vertices
Then we enter the two vertices – we can enter labels of vertices or their numbers. When asked
Forget values on lines answer [Yes] if searching for the shortest path according to lengths, and [No] if searching for the shortest path according to values of lines.

When asked
Identify vertices in source network answer [No].

The result is a new (sub)network, containing the two selected vertices, vertices that lie on the shortest path and corresponding lines. The resulting path can be drawn using
Layout/Energy/Kamada Kawai/Fix first and last

(first and last vertex should lie in the opposite corners).

Explanation: In most programs for data analysis (like SPSS) the result is obtained in some textual form. In contrast most operations in Pajek return as a result another object (new network, partition...) which can be further analysed.

The same is true when searching for the shortest paths – the result is a new network, that can be further analysed, visualized...

Be careful: in the resulting network the vertices are sequentially enumerated (with numbers from 1 on). If we want to 'recognize' which vertices from the primary network are on the shortest path we must choose marking of vertices using labels: (Options/Mark Vertices Using/Labels or Ctrl+L).

To find **chains** (where direction of lines are not important) in a directed network we can first transform directed lines to undirected using Network/Create New Network/Transform/Arcs to Edges/All.

Example: flow2.net: find the shortest path / chain between $v1$ and $v10$ according to both criteria

Be careful: the operations in Pajek are always executed on the selected network, therefore before running any operation we must check if the selected network is really the one we need.

We can find **all shortest paths** between two vertices using
Network/Create New Network/SubNetwork with Paths/...
...All Shortest Paths between Two Vertices
and answers to all questions as before (when searching for only one path).

When searching for a path between two vertices it can happen that the second vertex cannot be reached from the first – the path does not exist.

Diameter of the network can be found using

Network/Create New Network/SubNetwork with Paths/Info on Diameter

Pajek returns only the two vertices that are the furthest away. If we want to get the path too, the ordinary searching for the shortest paths between the two vertices must be run.

Be careful: Computing the diameter of the network is a very time consuming operation – it can be performed on smaller networks only.

Example: In the network of English words (dic28.net) find the shortest paths between:

white – yellow, engaged – skeptics, . . .

k-neighbors

Vertex j is a k-neighbor of vertex i if the shortest path from vertex i to vertex j has length k.

In flow2.net, the distances of all vertices from vertex $v1$ are:

vertex	1	2	3	4	5	6	7	8	9	10
k	0	1	2	2	1	1	2	3	3	3

In Pajek the distances of all vertices from selected vertex are computed using: Network/Create Partition/k-Neighbours/Output
We input the selected vertex (number or label) and when asked **Maximal distance** we input 0 – we want to check all distances.

The result of this operation is ***a partition*** – table: for every vertex we get the distance (number) from selected vertex.

Distances of vertices that cannot be reached from the selected vertex are set

to some large number (9999998).

Therefore: using Network/Create Partition/k-Neighbours/Output we compute how many steps we need to reach all other vertices from the selected vertex.

Similarly using Network/Create Partition/k-Neighbours/Input we compute how many steps we need to come from all other vertices to the selected vertex. Using Network/Create Partition/k-Neighbours/All we compute distances without taking directions of lines into account.

The result can be in the case of smaller networks displayed by double clicking the resulting partition or selecting the icon Edit/Partition.

Examples:
Find distances of all vertices from vertex $v1$ in flow2.net.
In network dic28.net find distances of all words from selected word.

In the case of larger networks we are not interested in distances of all vertices, but we want to see only some closest or the most distant vertices.

After computing distances from selected vertex, we can display 20 closest vertices and the frequency distribution of distances using: Partition/Info and when asked

Input 1 or 2 numbers: +/highest, -/lowest

input -20. If we want to display 20 the most distant vertices we input 20. When asked **Select minimum frequency** leave value 1 (in frequency distribution the class is shown if there is at least one unit in it).

Extracting k-neighbourhood of selected vertex

We can extract from a network a subnetwork with vertices that are on distance at most 2 from selected vertex in the following way. Select Operations/Network+Partition/Extract SubNetwork and input for the lower limit [0] and for the upper [2].

Before doing that we must select the network and corresponding partition in the main window of Pajek.

Result can be checked with the picture of the network.

General rule how to find commands in Pajek

Commands are put to menu according to the following criterion:
commands that need only a network as input are available in menu Net,
commands that need as input two networks are available in menu Networks,
commands that need as input two objects (e. g., network and partition) are available in menu Operations,
commands that need only a partition as input are available in menu Partition. . . .

Strongly and weakly connected components

In Pajek, strongly connected components are computed using Network/Create Partition/Components/Strong,
weakly using Network/Create Partition/Components/Strong. Result is represented by a *partition* – vertices that belong to the same component have the same number in the partition.

Example: stropic.net.

Biconnected components

To compute bicomponents use:Network/Create New Network/...
...with Bi-Connected Components stored as Relation Numbers

Biconnected components are stored to hierarchy (articulation points belong to several components, therefore bicomponents cannot be stored in a partition like strongly connected components). Nodes in the hierarchy represent biconnected components.

We can travel in a hierarchy like in *Windows Explorer*. Vertices that belong to selected bicomponent are displayed by double clicking with the left mouse button (or single clicking with right mouse button) the node in hierarchy.

A selected bicomponent is extracted using: Hierarchy/Extract Cluster and entering the number of the node in hierarchy.

A subnetwork determined by obtained cluster is extracted using Operations/Network+Cluster/Extract SubNetwork

Additionally, when computing bicomponents, Pajek returns a partition with articulation points (vertices in cluster 1 are not articulation points, all others are). The cluster number tells, into how many pieces a network will fall, when removing a given articulation point from the network.

Finally, since the command is stored in *Create New Network* also new network where Biconnected components are stored as relation numbers is generated. See details on Multiple Relation Networks later.

Examples: aho1.net, usair.net.

Example: Airlines connections network

An airlines connections network in the USA (usair97.net) consists of 14 biconnected components of size at least 3. The largest bicomponent contains 244 airports, some of the smallest contain 3 airports. There exist 27 articulation points (if we compute bi-components of size 2 or more). The most important articulation point (airport) is *Dallas*. If we remove that vertex (airport is closed), the airlines connections network will fall to 14 disconnected pieces. The first 4 airports according to 'articulation point' criterion are (result obtained using Partition/Info):

```
Rank Unit Class Id
-------------------------------------
1     261    14 Dallas/Fort Worth Intl
2      13     7 Bethel
3       8     7 Anchorage Intl
4     201     6 San Francisco Intl
```

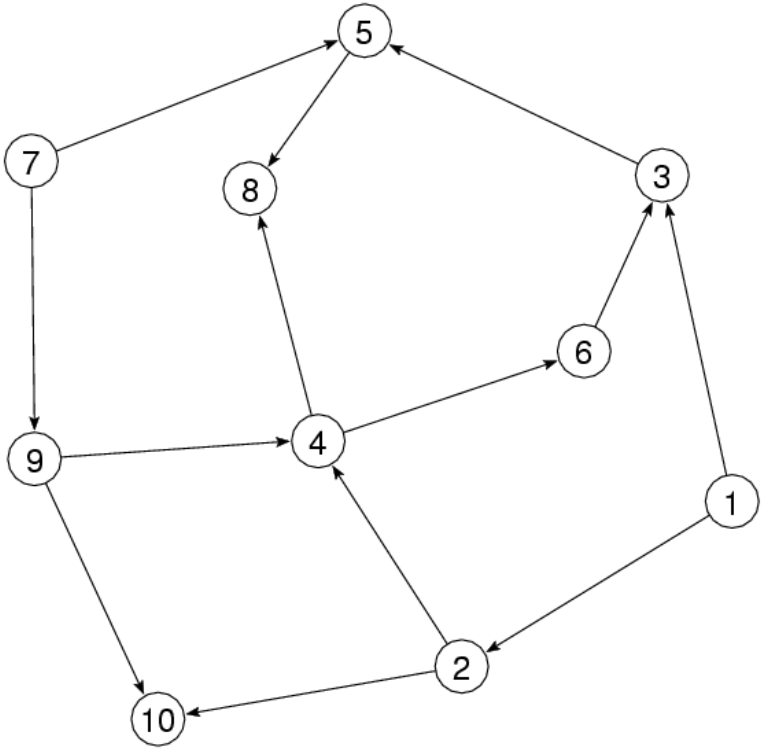
Acyclic networks

Examples of acyclic networks: project planning (critical path method – CPM), citation networks, genealogies...

In an acyclic network ***first vertices*** exist – vertices into which no line is coming. There exist ***last vertices*** too – vertices from which no line is going out.

For every vertex of an acyclic network the ***depth*** can be computed: We assign depth 1 to all first vertices and remove from the network first vertices and corresponding lines. In this way we obtain a new acyclic network. We assign depth 2 to all first vertices in the obtained network, remove first vertices and corresponding lines and continue the procedure.

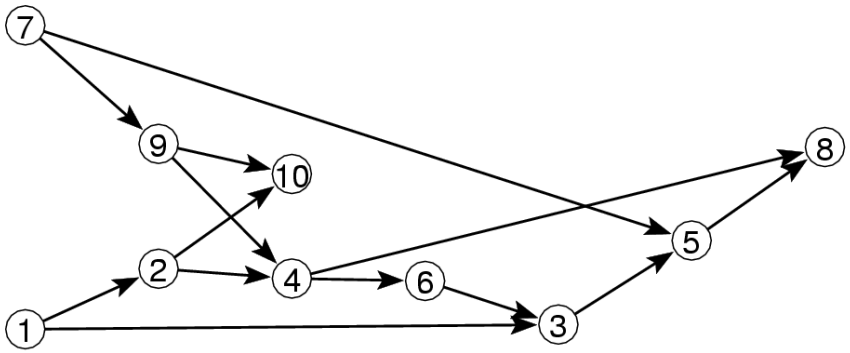
Example: wirth.net



vertex	depth
1	1
2	2
3	5
4	3
5	6
6	4
7	1
8	7
9	2
10	3

Depths can be used for drawing the acyclic network in **layers**.

If we use layers in x direction we get:



In Pajek depths are computed using
Network/Acyclic Network/Depth Partition/Acyclic

If the command Draw/Network+Partition is selected afterwards, each vertex is colored using a color that corresponds to its depth. Colors used are shown in Options/Colors/Partition Colors/for Vertices in Draw window.

Layout in layers is obtained using Layers/in y direction. If we want to improve the picture by hand, but we want to keep the layers, we can define the y coordinate as fixed using Move/Fix y
The y coordinate of any vertex cannot be changed afterwards.

Some useful commands

If we want to position vertices on a rectangular net or concentric circles we use

Move/Grid or Move/Circles

We input the size of the net or the density of concentric circles. The selected vertex will always 'jump' to closest position when moving the vertex.

Example: net.net.

If we have a network with values of lines we can use Options/Values of Lines to determine if the values of lines are similarities, dissimilarities, or we can forget them. The information is used when drawing using energy: vertices connected with larger values will be drawn close to each other in the case of similarities and far away in the case of dissimilarities.

Using Options/Interrupt we define how long (in seconds) the layout is optimized when using energy algorithms.

Using Options/ScrollBar On/Off the scrollbar is set into the Draw window. The scrollbar works in two different ways:

- if complete layout is selected, the scrollbar is used to rotate the picture,
- if part of the layout is selected (Zoom), the scrollbar is used to move the 'visible' frame.

When exporting layouts to EPS, SVG or VRML several parameters can be set in Export/Options. For explanation see Pajek wiki page.

In this way we can define in detail how the picture should look like. EPS pictures can be displayed using program GsView.

Examples

1. Find shortest paths between $v1$ and $v10$ in flow2.net
2. In the network of English words (dic28.net) find the shortest paths between:
   white – yellow
   engaged – skeptics
3. Find distances of all vertices from vertex $v5$ in flow2.net.
4. In dic28.net find the distances of all words from yellow. Extract and draw a subnetwork including all words that are at distance 3 or less from yellow (including yellow too).

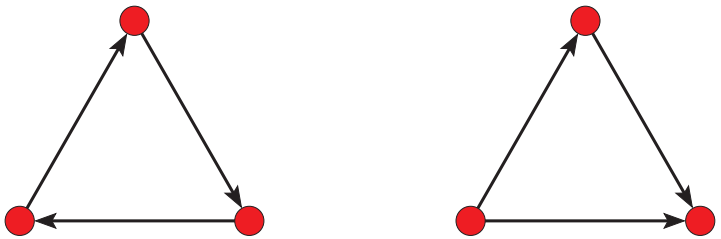
Short cycles

Network/Create New Network/with Ring Counts stored as Line Values/3-Rings

Network/Create New Network/with Ring Counts stored as Line Values/4-Rings

with suboptions for undirected and directed rings are available:

- **3-Rings** – For each line count number of 3-rings to which the line belongs.
  - **Undirected** – for undirected networks – count undirected 3-rings.
  - **Directed** – for directed networks – count **cyclic**, **transitive**, or all 3-rings, or count how many times each line is a transitive shortcut).



- **4-Rings** – For each line count number of 4-rings to which the line belongs.
  - **Undirected** – for undirected networks – count undirected 4-rings.
  - **Directed** – for directed networks – count **cyc**lic, **tra**nsitive, **gen**ealogical, **dia**mond, or all 4-rings, or count how many times each line is a transitive shortcut.

Example: count undirected 3 and 4-rings in write.net

Centrality measures in Pajek

Degree

In Pajek degrees are computed using Network/Create Partition/Degree or Network/Create Vector/Centrality/Degree
and selecting Input, Output or All. Using first command we get a Partition as a result using second command we get a Vector: Vertices with the highest degree can be displayed using Vector/Info. For smaller networks the result can be displayed by double clicking the partition window, selecting File/Vector/Edit or selecting the corresponding icon.

For weighted degrees (taking line values into accounts, use: Network/Create Vector/Centrality/Weighted Degree.

Closeness and betweenness

Also other entralities can be found in Network/Create Vector/Centrality.

When computing centrality according to degree and closeness we must additionally select Input, Output or All. If network is undirected we can choose Input or Output (the result is the same).

In the other window the network centralization index is given.
List of the selected number of most central vertices can be obtained by Vector/Info.
Centrality measure of each vertex obtained (real number between 0 and 1) can be used to determine the size of a vertex in a layout, if we select Draw/Network+Vector. Examples usair.net, sampson.net, flor.net.

Exporting from Pajek to statistical packages R and SPSS

Tools/ R/ Locate R to find the executable file `Rgui.exe`, which is usually installed on directory `c:\Program Files\R\rw????\bin`
then **Tools/ R/ Send to R** to send vectors or networks to R.

In R or SPSS we can perform further statistical analysis of objects obtained from Pajek.

Pajek and R

Networks and vectors can be exported from Pajek to statistical package R (and vice versa). R can be downloaded for free from
`htpp://cran.r-project.org`

In R we can perform ordinary statistical analysis of networks and vectors.

For example: export vectors containing closeness and betweenness central-

ity to R, and

- draw scatterplot,
- draw histogram,
- compute correlation between the two measures.

Export to SPSS

To connect Pajek with SPSS use **Tools/ SPSS/ Locate SPSS** to find the executable file `runsyntx.exe`, which is usually installed on directory `c:\Program Files\SPSS`
then **Tools/ SPSS/ Send to SPSS** to send partitions, vectors or networks to SPSS.

Export to Tab delimited File

Export all or selected partitions and vectors to a file that can be later directly read by other statistical software like Stata, or spredsheets like Excel...

Measures of prestige in Pajek

Look at another two measures of prestige, which are especially useful in the case of WWW (directed network of home pages).

In directed networks we can usually identify two types of important vertices: hubs and authorities. Each home page describes something (is an *authority*) and because of that other pages point to it. But on the other hand each page points to some other pages (is a *hub*).

Vertex is a *good hub*, if it points to many *good authorities*, and is a *good authority*, if it is pointed to by many *good hubs*.

See: Kleinberg, Jon M.: Authoritative Sources in a Hyperlinked Environment. *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms.* Edited by Howard Karloff (SIAM/ACM-SIGACT, 1998).

Hubs and authorities are computed in
Network/Create Vector/Centrality/Hubs-Authorities

We get additional question: How many hubs and how many authorities are marked in the partition. Results are:

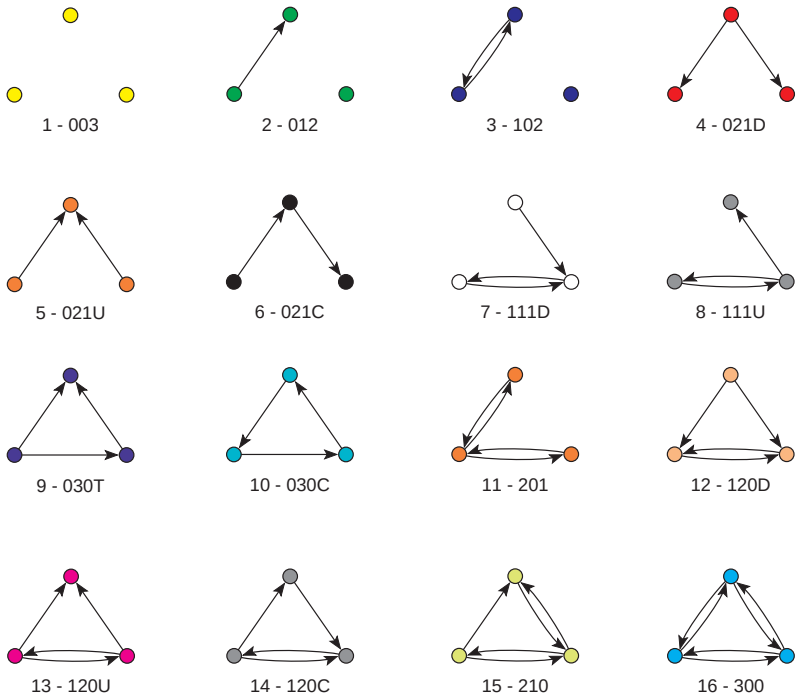
- Partition, where value 1 (yellow) means, that the vertex is a good authority, value 2 (green) means, that the vertex is a good authority and a good hub, and value 3 (red) means, that the vertex is a good hub.
- Vector with *Hub Weights*, larger value means better hub.
- Vector with *Authority Weights*, larger value means better authority.

Be careful: Algorithm supposes that values on lines represent similarities (larger value means more important choice).

Example: football.net – export of football players (SVG).

Triadic census

To get triadic census run: Network/Info/Triadic Census. Example: exclique.net.



Cliques

Subset of vertices in a network is called a ***clique***, if every vertex from the subset is connected to all other vertices in the subset (clique is a special type of a ***core***). A clique (in social networks) represent a subset of persons who are connected as much as possible.

Searching for cliques is computationally much more expensive than searching for cores. Therefore we will only search for cliques of size 3 or 4 at most in smaller networks.

3-clique is triad number 16. Using counting of triads we can find how many 3-cliques exist in a network.

But if we want to find all occurrences of 3-cliques we must use the general procedure for searching fragments in network.

It was successfully applied to searching for relinking marriages in genealogies.

Pattern (fragment) searching

The procedure finds any interesting smaller network (fragment) in a larger network. In our case the fragment is a clique on 3 vertices.

The 3-clique can be generated in Draw window like explained in the beginning (interactive definition of networks).

But it is easier to generate a directed complete network on 3 vertices:
Network/Create New Network/Complete Network/Directed
We define this network as a fragment by selecting it as the first network. Then we put the original network in the second box.

Be careful: Algorithm for searching fragments *distinguishes* between edges and bidirected arcs. If we have some edges in the network, we must transform them to bidirected arcs first using
Network/Create New Network/Transform/Edges $-$ > Arcs.

After we selected fragment (first network) and original network, where we want to find such fragments (second network), we run searching for fragments using
Networks/Fragment (First in Second)/Find.

We get three objects as a result:

1. Subnetwork containing only all occurrences of fragments (cliques).
2. Partition with values 0 and higher: value 0 means that the correspondent vertex does not belong to any fragment (clique), value a means that vertex belongs to a fragments (cliques).
3. Hierarchy with all fragments.

Examples: find 3 and 4 cliques in advice.net, exclique.net.

Pattern can be any connected subgraph. Find for example all 5 or 6 rings in 1crn.net

Cores

A subset of vertices is called a k**-core** if every vertex from the subset is connected to at least k vertices from the same subset.

Cores in Pajek can be computed using Network/Create Partition/k-Core and selecting Input, Output or All core. Result is a partition: for every vertex its core number is given.

In most cases we are interested in the highest core(s) only. The corresponding subnetwork can be extracted using Operations/Network+Partition/Extract SubNetwork and typing the lower and upper limit for the core number. (if we want to extract the highest core only, we type the same core number twice).

Examples:

Compute and extract the highest core in network write.net.

The network is undirected. *The highest core is 4-core.* (SVG picture).

Compute cores in usair.net.

Computing and extracting a core of a network is one of the possibilities to determine ***the boundary*** of the network: Often, we are interested only in the 'densest' part of a large network. In this case we compute cores, and extract only the part belonging to some core number or higher.

Take the relation *whom would you ask for the advice* (directed network):

For people belonging to an input k-core, it holds that at least k people from the k-core would ask any of the persons belonging to k-core.

For people belonging to output k-core, it holds, that every person from k-core would ask for the advice at least k persons from k-core.

Take the network advice.net: *whom would you ask for the advice*. The network is directed.

The frequency distribution of ***input*** k-cores (the table is obtained using Pajek command Partition/Info):

Class	Freq	Class*Freq	Represent.
5	2	10	I15
8	1	8	I2
11	26	286	I1

Explanation

In the network an 11-core exists with 26 students in it: there exist 26 students, among which every one would be asked for advice from at least 11 others from this 26.
One of these 26 students is also student `I1`.

The frequency distribution of ***output*** k-cores:

Class	Freq	Class*Freq	Represent.
1	2	2	I2
2	2	4	I3
4	1	4	I7
7	2	14	I11
8	2	16	I6
9	20	180	I1

Explanation

In the network a 9-core exists, with 20 students in it: there exist 20 students, everyone of whom would ask for the advice at least 9 others from this 20. One if this 20 students is student `I1`.

Islands

In a network where weights on vertices are given we can find *vertex islands*. Similarly, if line values are given *line islands* can be found.

Properties should not be given in advance – they can be computed as well (centrality measures, core numbers, rings counts. . .).

Line islands are clusters of vertices, such that inside clusters line values are higher than between clusters. In Pajek line islands are computed using Network/Create Partition/Islands/Line Weights

For vertex islands run: Operations/Network+Vector/Islands/Vertex Weights

We must also select the minimum and the maximum size of island allowed.

Compute line islands on network write.net with 3-rings for line weights. Try different criteria for island sizes (e.g. min=3, max=12).

Compute vertex islands on network write.net with core numbers for weights on vertices. Try different criteria for island sizes (e.g. min=3, max=12).

Community detection methods

Communities - dense clusters for which there are more lines inside than among clusters (values of lines are taken into account too).

In Pajek two community detection methods are available: *Louvain method* and *VOS Clustering*.

When applying Louvain method we search for partition into clusters with the highest value of *modularity (Q)*. Modularity is defined in the following way:

$$Q = \frac{1}{2m} \sum_s (e_s - r * \frac{K_s^2}{2m})$$

- m – total number of lines in network,
- s – cluster (community),
- $e_s = \sum_{ij \in s} A_{ij}$ – 2 times the number of lines in community s

- $K_s = \sum_{i \in s} k_i$ – sum of degrees in community s
- r – *resolution parameter*, default value 1 means modularity as originally defined

Similar method is *VOS Clustering*, where *VOS quality function* is taken into account instead of modularity.

By changing *resolution parameter* - r we can get larger or smaller communities. By default resolution parameter is set to 1. Setting r larger than 1 means searching for larger number of smaller communities. Setting r smaller than 1 means searching for smaller number of larger communities.

Examples: football.net, import.net.

Some more examples:
http://mrvar.fdv.uni-lj.si/pajek/community/LouvainVOS.htm

Two-mode networks in Pajek

A two-mode network is defined on an input file in the following way (Davis.net):

```
*Vertices 32 18
 1 EVELYN           27 E9
 2 LAURA            28 E10
 3 THERESA          29 E11
 4 BRENDA           30 E12
 5 CHARLOTTE        31 E13
 6 FRANCES          32 E14
 7 ELEANOR          *Edgeslist
 8 PEARL            1 19 20 21 22 23 24 26 27
 9 RUTH             2 19 20 21 23 24 25 26
10 VERNE            3 20 21 22 23 24 25 26 27
11 MYRNA            4 19 21 22 23 24 25 26
12 KATHERINE        5 21 22 23 25
13 SYLVIA           6 21 23 24 26
14 NORA             7 23 24 25 26
15 HELEN            8 24 26 27
16 DOROTHY          9 23 25 26 27
17 OLIVIA           10 25 26 27 30
18 FLORA            11 26 27 28 30
```

```
19 E1                12 26 27 28 30 31 32
20 E2                13 25 26 27 28 30 31 32
21 E3                14 24 25 27 28 29 30 31 32
22 E4                15 25 26 28 29 30 31 32
23 E5                16 26 27 28 30
24 E6                17 27 29
25 E7                18 27 29
26 E8
```

Explanation

The only difference comparing to ordinary networks is that we have to specify the total number of vertices (in our case 32) and number of vertices that belong to the first subset (in our case 18 women). First, all vertices from the first subset must be listed (women) and afterwards all vertices from the second subset (events). Partition into the two subsets is obtained using Network/2-Mode Network/Partition into 2 Modes, where value 1 is given to vertices from the first subset (women), and value 2 to vertices from the second subset (events).

Transforming to valued networks

The network is transformed into an ordinary network, where the vertices are elements from the first subset (in our case women), using Network/2-Mode Network/2-Mode to 1-Mode/Rows.
If we want to get a network with elements from the second subset we use Network/2-Mode Network/2-Mode to 1-Mode/Cols.
Network with or without loops can be generated:
Network/2-Mode Network/2-Mode to 1-Mode/Include Loops.

We can generate network with values on lines (in our case number of common events) or network with multiple lines – for each common event a line between corresponding two women:
Network/2-Mode Network/2-Mode to 1-Mode/Multiple Lines

For our purposes we will always generate a valued network, sometime the loops will be useful sometimes not.

Obtaining picture of valued network

We store values of loops (e.g. total number of events for each women) into vector using Network/Create Vector/Get Loops and use later this vector to determine sizes of vertices (Draw/Network+Vector).
After network with values on lines is generated we can visualize it in different ways:

1. picture of complete network: First we draw network using Energy drawing, providing that option Options/Values of Lines/Similarities is selected (vertices connected with higher values will be drawn closer).

Values of lines can be shown using different widths of lines (Options/Lines/Different Widths) and/or greyscale (Options/Lines/GreyScale).

After that we export picture to SVG, where we can interactively add lines according to their values:
Export/SVG/LineValues/Nested Classes.

2. picture of the most important part of the network: only lines with values which are large enough are kept.

Distribution of line values can obtained using
Network/Info/Line Values.

According to the distribution, we remove lines with low values from the network using
Network/Create New Network/Transform/Remove/lines with value/lower than
and entering the threshold value.

Islands

In a network where some properties (values) of vertices or lines are known we can find *islands*. Islands are called *vertex islands* if values of vertices are given, and they are called *line islands* if values of lines are given.

Lets take the network obtained from two mode network. As we know, in such network values of lines are given, therefore we can find line islands – clusters of vertices, connected with lines having higher values than values of lines going out (inside islands line values are higher than between islands).

We must also select the smallest and the largest size of island allowed. In Pajek line islands are computed using

Network/Create Partition/Islands/Line Weights

Example: Davis.net – compute line islands of size 2 to 6 for both networks obtained from a two mode network

Direct analysis of 2-mode networks

Compute 4-rings on two mode network using
Network/Create New Network/with Ring Counts stored as Line Values/4-Rings

As result weights on lines are obtained (remove lines with low values, compute line islands...)

To get overview of 2-mode cores run
Network/2-Mode Network/Core/2-Mode Review.

To get just the border values which are the most interesting use
Network/2-Mode Network/Core/2-Mode Border.

When you decide which core is the most interesting compute it using Network/2-Mode Network/Core/2-Mode. by providing minimum degree in first and in second mode. Result is a partition with value one for vertices belonging to the core and 0 otherwise.

Hierarchical clustering in Pajek

Procedure is composed of two steps:

- computing dissimilarity matrix
- hierarchical clustering according to the obtained dissimilarity matrix

Before running procedure generate complete cluster using
Cluster/Create Complete Cluster.
In this way dissimilarities will be computed among all units.

Run Operations/Network+Cluster/Dissimilarity/Network based and select

- **d1/All** – if you want to consider network as binary matrix, or
- **Corrected Euclidean** or **Corrected Manhattan** distance – for valued networks, in this case parameter p (0, 1, or 2) must be entered. Parameter p tells how to count *diagonal* and *direct connection* between two units.

Additionally you are asked for the name of file where EPS picture of dendrogram will be saved (you can use program **GSView** to see the result). If you press *Cancel* at this point only dissimilarity matrix will be computed (the procedure will not continue with hierarchical clustering).

Results of hierarchical clustering procedure in Pajek are:

- Dissimilarity matrix.
- EPS picture of dendrogram.
- Permutation of vertices according to dendrogram. You can use this permutation to draw reordered matrix in EPS (File/Network/Export Matrix to EPS/Using Permutation)
- Hierarchy representing hierarchical clustering.
  Some approaches to use this result for further analysis:
  - Check the option Edit/Show Subtree (show all units in the subtree of selected cluster).

- After you decide what are suitable clusters (according to dendrogram), close the corresponding nodes by pressing Edit/Change Type or Ctrl+T so long that the word **Close** appears.
- Transform hierarchy into partition (Hierarchy/Make Partition).
- You can examine the result using Draw/Network+Partition, or draw the reordered matrix File/Network/Export Matrix to EPS/Using Permutation with lines among clusters.

By default *Ward* method is used. If you want to use another method, select Network/Create Hierarchy/Clustering*/Options

Examples: shr1.net (2 clusters, 3 clusters, clusters with at least 4 vertices) import.net (2 clusters), sampson.net (3 clusters).

Blockmodeling in Pajek

Blockmodeling can be run in two different ways:

- start with random partition into given number of clusters
  Network/Create Partition/Blockmodeling*/Random Start
- optimize the given partition (for example partition obtained from hierarchical clustering)
  Network/Create Partition/Blockmodeling*/Optimize Partition

After running the appropriate option, select

- type of equivalence – Structural or Regular. You can also define your own blocks, by selecting 3–Define. In this case you can define for each block in the image matrix which type of blocks are allowed and what is the penalty for the block. The model defined in this way can be saved to a file (Save as MDL File) and loaded later (Load from MDL File).
- number of repetitions

- number of clusters

After you set all options, press Run.

In the case of optimization of given partition, you are not asked for number of clusters, and number of repetitions.

The result of blockmodeling procedure are all different partitions with the lowest value of criterion function (one or more partitions).

If you want to export reordered matrix (with lines among clusters) you must first generate the permutation using Partition/Make Permutation.

Compute structural and regular equivalence for sampson.net into 2 and 3 clusters.

Example - prespecified blockmodel

In the case of the student network (class1.net) we can consider a core-periphery model: there are diligent students (the core group) with good learning materials and there are less motivated students (the periphery group) who borrow studying material from the students in core group and not from students in their own group. Therefore, our pre-specified blockmodel is:

	C_1	C_2
C_1	com, reg	-
C_2	com, reg	-

In the table `com` stands for complete block, `reg` stands for regular block, and − stands for null block.

Try the above model!

Multiple relations networks

Networks containing multiple relations on the same set of vertices, e.g. Sampson monastery data. We define the relation number and corresponding label by extending *Matrix, *Arcs, or *Arcslist statement:

```
*Matrix :1 "Liking1"
*Matrix :2 "Liking2"
*Arcs :1 "Liking1"
*Arcs :2 "Liking2"
```

See sampsonmul.net

Network/Multiple Relations Network/Info – number of arcs/edges in each relation.

Network/Multiple Relations Network/Extract Relation(s) – extracting selected relations from multiple relations network.

Visualization of multiple relations networks:

Options/Colors/Edges/Relation Number and
Options/Colors/Arcs/Relation Number - colors of edges/arcs are determined by relation number

Options/Colors/Relation Colors - color table for relation numbers.

Options/Lines/Draw Lines/Relations - showing/hiding lines belonging to selected relations e.g. 1-3,6,10-15.

Temporal networks

Networks changing over time: In temporal network vertices and lines are not necessarily present or active in all time points.

Description in input file:

t_i – in time point t_i,
t_i–t_j – from time point t_i to time point t_j,
t_i–$*$ – from time point t_i on.

```
*Vertices 3
1 "a" [5-10,12-14]
2 "b" [1-3,7]
3 "e" [4-*]
*Edges
1 2 1 [7]
1 3 1 [6-8]
```

Pajek: Read temporal network and then
Network/Temporal Network/Generate in Time
to obtain separate networks in different time points.

lin_rel4.net: Multiple relation temporal network – actors and relations among them in the long-running German soap opera called 'Lindenstrasse'.

For each actor her/his name, gender, birthdate, and several other records are available. Additionally for each actor episode numbers in which the actor played actively are given.

For each line in the network its meaning is given: family relation, business relation, unfriendly relationships, ...

Properties of vertices are represented by different shapes, colors, sizes of vertices: e.g. triangles correspond to men, circles to women; and properties of lines are represented by colors: green line stands for family relation, blue for business relations, ...

Generate only different networks in time points 1 to 50.