Cluster

One of the important goals of network analysis is to find *clusters* of units, which have similar (equal) structural characteristics, which are determined using relation R.

Cluster – C is a subset of units.

In Pajek default extension for cluster is .CLS.

If we are interested only in units 2, 4 and 5 in the test network, the cluster in input file is described in the following way:

```
2
4
5
```

Smaller clusters can be built using Pajek too, in the following way:
First create an empty cluster (cluster without units) using
Cluster/Create Empty Cluster
then select File/Cluster/Edit or the corresponding icon and by double clicking on Add new vertex
add new units to the cluster.

Partition of units

Clusters form a *partition* of the set if units U

$$\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$$

if:

$$\bigcup_i C_i = U \qquad \text{and} \qquad i \neq j \Rightarrow C_i \cap C_j = \emptyset.$$

We can imagine partition as a one dimensional table of equal dimension as is the number of units in network. i-th element in the table tells, to which cluster unit i belongs. Clusters are in this case presented as integer numbers. Sometimes we put 'not interesting units' in cluster 0.

We need partition whenever we want to describe a property of unit using some integer number. So far we have already used partitions for

- determining distances of vertices from a selected vertex – k-neighbours
- determining depths of vertices in an acyclic network
- determining degrees of vertices

If vertices represent people, partition can be used to describe their gender (1-men, 2-women).

Default extension for partition is .CLU – clustering.

If we have a network with 5 vertices, and want to set vertices 1 and 5 into cluster 1, 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 built Partition using Pajek too, so that we first use Partition/Create Null Partition
The dimension of partition is by default set to the number of vertices in selected network, what is OK. 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/Draw-Partition or press Ctrl+P colors of vertices will be used to show to which cluster each vertex belongs.

The second possibility to determine partition is:

- Select Draw/Draw-SelectAll or press Ctrl+A. Using that command three operations are executed: new partition of equal dimension as is 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 network: By pressing the middle mouse button we increment the cluster number of selected vertex (if we hold the Alt key on keyboard at the same time, we decrement the cluster number).

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

Description of cluster numbers and corresponding colors is given in http://vlado.fmf.uni-lj.si/pub/networks/pajek/part_col.pdf

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

More than in cluster or partition of vertices itself we are interested in the subnetwork (vertices and lines), which the vertices in selected cluster induce.

In Pajek we extract subnetwork determined by a given cluster using:
Operations/Extract from Network/Cluster

Before we run this operation we must select the network and cluster in the main window.

We will extract selected cluster which is determined by a partition even more often. In this case we use:
Operations/Extract from Network/Partition

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.

Components

Three types of components will be defined: *strong*, *weak* and *biconnected*.

Subset of vertices in network is called ***strongly connected component***, if we can (by taking directions of lines into account) from every vertex of the subset reach every other vertex belonging to the same subset.

If direction of lines is not important (we consider network to be undirected), such a subset is called ***weakly connected component***.

Example: Let vertices of network correspond to buildings in the city, lines correspond to streets that connect the buildings. Some streets are ordinary (undirected lines), some are one way only (directed lines).

All buildings that can be reached from one to another using the car, belongs to the same strongly connected component (we must take one way streets into account).

All buildings that can be reached from one to another by walking, belongs to the same weakly connected component (one way streets are no limit).

Lets take the relation *whom would you invite to the party*. For all persons belonging to the same strongly connected component, it holds:

- everybody will (at least indirectly) invite everybody else from the same strongly connected component
- everybody will be (at least indirectly) invited by everybody else from the same strongly connected component

If network is undirected, strongly connected component are the same as weakly connected, so they are simply called connected components.

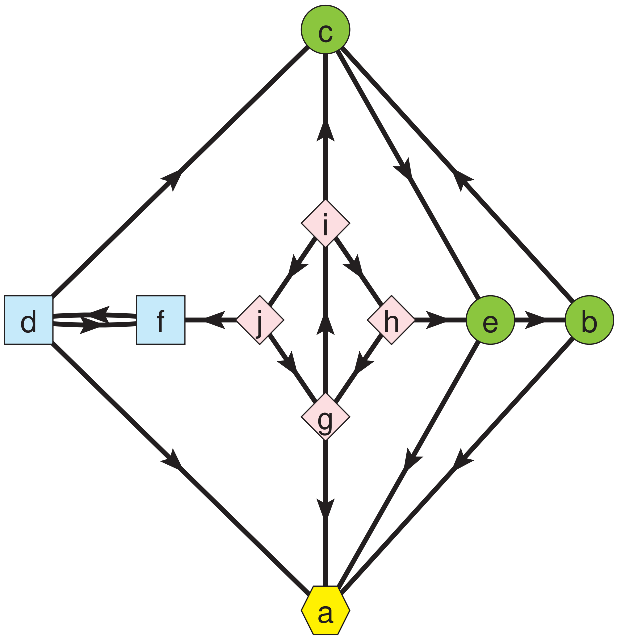
Connection between ***walks*** in network and ***components***

Between any two vertices from the same strongly connected component there always exists a *walk*. We also say that the two vertices are strongly connected.

Between any two vertices from the same weakly connected component there always exists a *chain*. We also say that the two vertices are weakly connected.

Network is called ***strongly/weakly connected***, if every pair of vertices is strongly/weakly connected.

Example



Network is weakly connected.

There exist 4 strongly connected components:

1. (a)
2. (b, c, e)
3. (d, f)
4. (g, h, i, j)

Vertices that belong to the same strongly connected components are drawn using the same shape.

Shape of vertex in Pajek is determined in input file, so that after vertex label (and coordinates) we add:

box, ellipse, diamond or triangle:

```
*Vertices 4
1 "a" box
2 "b" ellipse
3 č" diamond
4 "d" triangle
...............
```

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

Find connected components in the dictionary of English words (dic28.net)!

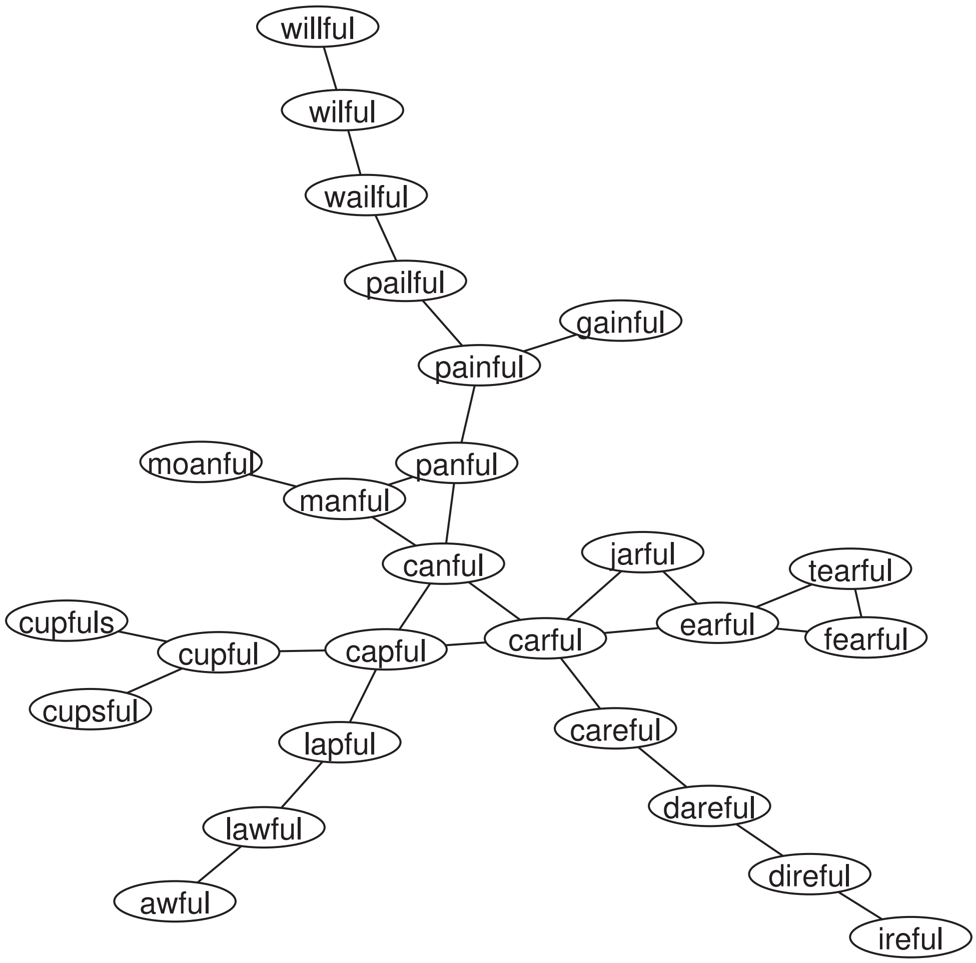
What is the number of components? ________ 17903

Number of vertices in the largest component? ________ 24831

Size of the smallest component? ________ 1 (isolated vertex)

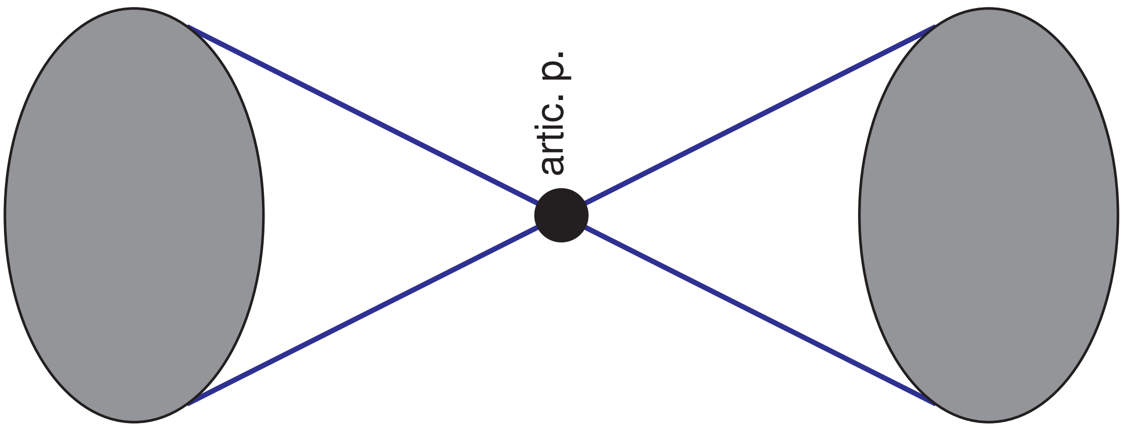
Extract and draw one of the smallest components.

One of the smallest components:



Biconnected components

Lets take the undirected connected network. Vertex a of the network is ***articulation point*** of the network, if there exist two other, different vertices v and w, so that, every chain between the two vertices includes also vertex a. Simply saying: vertex a is articulation point, if removing the vertex from the network causes the network to become disconnected.

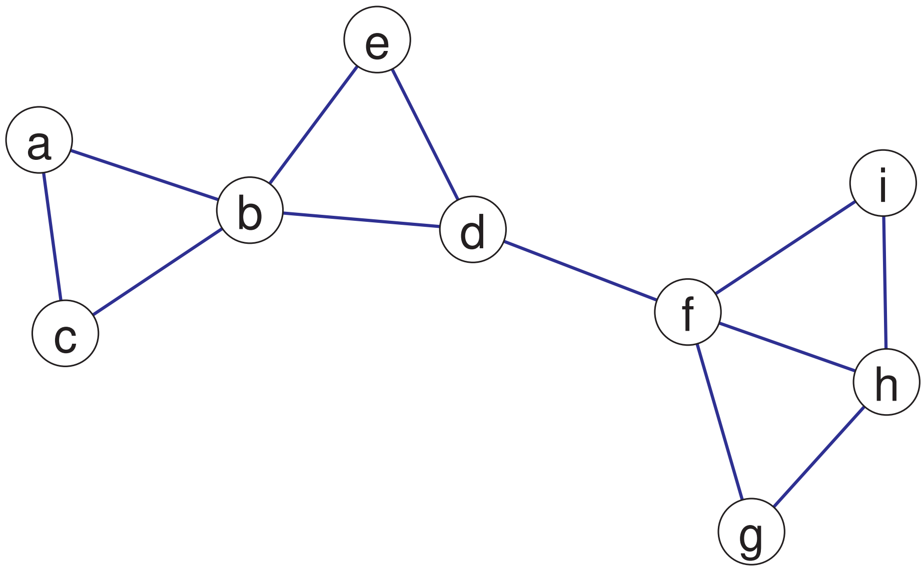


Network is called *biconnected*, if for every triple of vertices a, v and w there exists chain between v and w, which does not include vertex a.

Simply: network is called *biconnected*, if after removing any vertex, network remains connected.

Biconnected network does not contain any articulation point.

Example

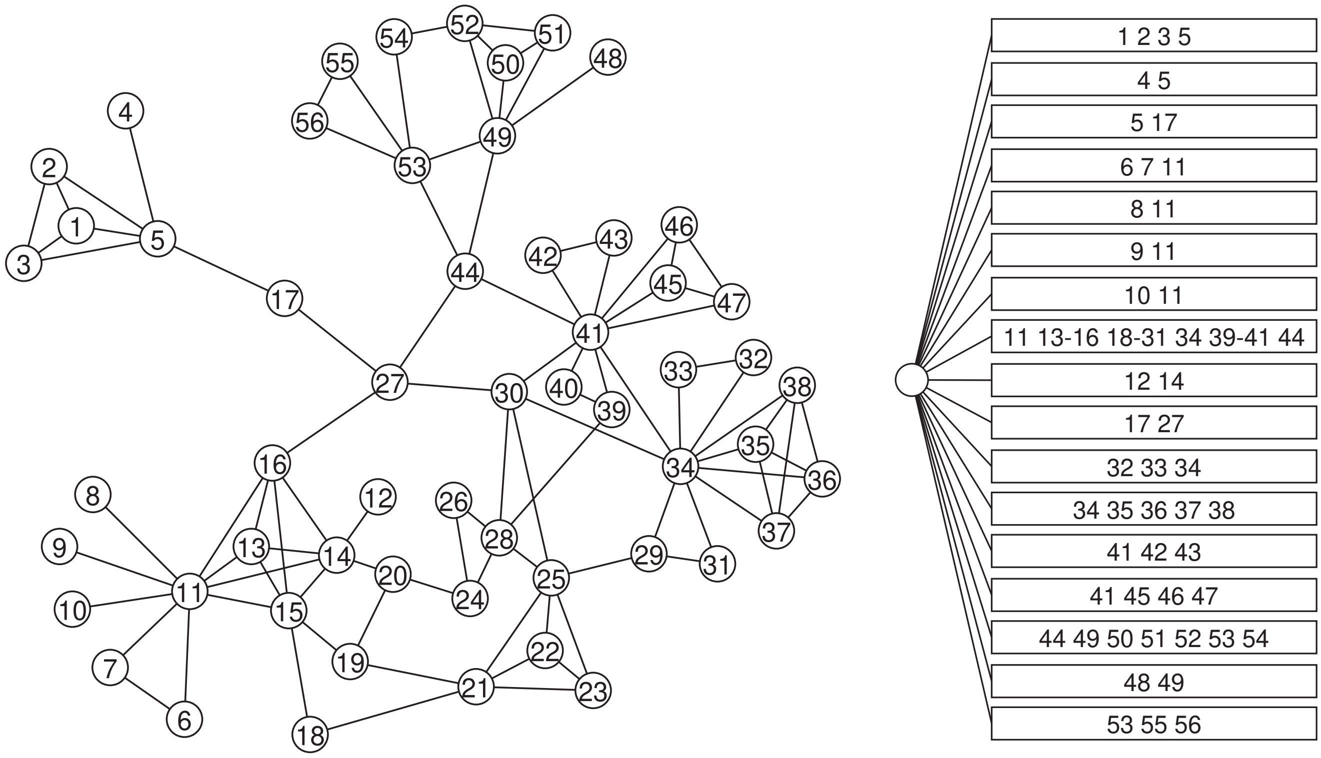


Network in the picture (aho1.net) consists of 4 biconnected components:

- (a, b, c)
- (b, d, e)
- (d, f)
- (f, g, h, i)

Articulation points are b, d and f.

Network that consists of 17 biconnected components:



Definition of components using walks

For ***strongly connected*** components ***direction of lines is important***:
Subset of vertices is called strongly connected component, if (when taking directions of lines into account) there exists *at least one walk* from any vertex to any other vertex from the subset.

For ***weakly connected*** and ***biconnected*** components ***direction of lines is not important***:
Subset of vertices is called weakly connected component, if there exists *at least one chain* from any vertex to any other vertex from the subset.

Subset of vertices is called biconnected component, if there exist *at least two chains* from any vertex to any other vertex from the subset.

In some occasions it is welcome that network is 'well' connected (large strongly, weakly and biconnected components). Such examples are communication networks (streets, information flow, ...); sometimes it is not (e. g. infections).

Biconnected components in Pajek

To compute bicomponents use: Net/Components/Bi-components

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

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

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

Subnetwork, determined by obtained cluster is extracted using Operations/Extract from Network/Cluster

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, to how many pieces network will fall, when removing given articulation point from the network.

Example: Airlines connections network

Airlines connections network in USA (usair97.net) consists of 14 biconnected components of size at least 3. The largest component contains 244 airports, some smallest contain 3 airports. There exist 27 articulation points. The most important articulation point (airport) is *Dallas*. If we remove that vertex (airport is closed), the airlines connections network will fall to 14 not connected pieces. First 8 airports according to 'articulation point' criterion are (result obtained using Info/Partition):

```
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
5     152     6 Pittsburgh Intll
6     182     5 Lambert-St Louis Intl
7     258     4 Phoenix Sky Harbor Intl
8      67     4 Minneapolis-St Paul Intl
```

Cores

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

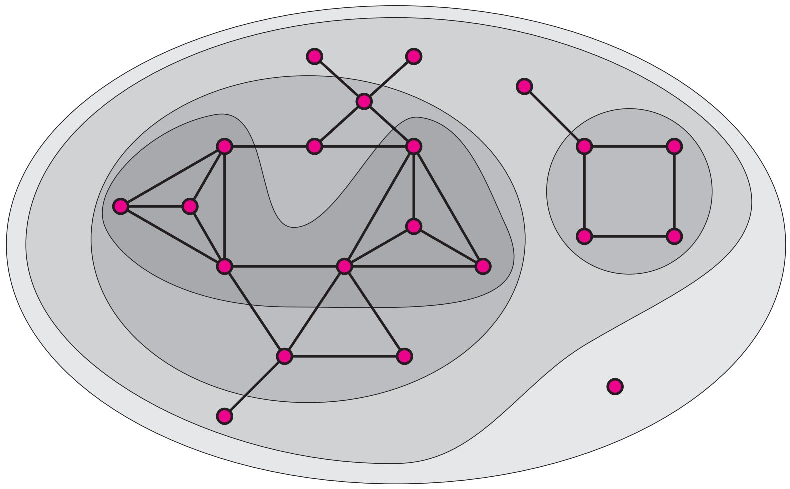
Formally:
Let $N = (V, L)$, $L \subseteq V \times V$ be a network.

A maximal subgraph $H = (W, \cap W \times W)$ induced by the set W is a k-*core* iff $\forall v \in W : \deg_H(v) \geq k$.

Special example:
Subset of vertices is called ***clique***, if every vertex from the subset is connected to every other vertex from the subset.

Computing cliques is very time consuming, while computing cores is not.

0, 1, 2 and 3 core:



Cores can be computed according to lines going into vertices (Input), lines going out of vertices (Output) or all lines (All). In the case of undirected networks input and output cores are the same.

Properties of cores:

- Cores are nested (see figure):

$$i < j \Longrightarrow H_j \subseteq H_i$$

- They are not necessarily connected subnetworks (see figure).
- Cores can be generalized to networks with values on lines.

Cores in Pajek can be computed using Net/Partitions/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/Extract from Network/Partition 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).

Example:
Compute and extract the highest core in network write.net. Network is undirected. *The highest core is 4-core.*

Computing and extracting a core of a network is one of possibilities to determine ***boundary*** of the network: Often, we are interested in the 'densest' part of a large network only. 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 input k-core, it holds, that at least k people from 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*. Network is directed.

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

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

Explanation

In network 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 this 26 students is also student I1.

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

Example

What is the highest core in network of airline connections in USA – usair97.net (network is undirected)? Which airports belong to the highest core?

Examples

1. Find strongly connected components in stropic.net. Extract the largest component.
2. Find connected components in dic28.net!
   What is the number of components? ____________17903
   The largest component has ____________24831 vertices
   The smallest component(s) have ____________1 vertex
   Extract and draw one of the smallest component (component with 15-40 vertices).
3. Find biconnected components in aho1.net. Extract the largest component as a new network.
4. Find biconnected components in write.net. Extract the largest component as a new network.
5. Find biconnected components in airlines connections network (usair97.net). Extract the component with 6 airports. Find first 8 airports which are the most crucial – articulation points. Draw the picture of network, so that the size of vertices will represent to how many pieces the network will fall if given airport is closed.
6. Compute and extract the highest core in write.net.
7. Compute and extract the highest input and output core in

advice.net.

8. What is the highest core in airlines connections network (usair97.net)? Which airports belong to that core?