Cluster

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

Cluster – C is a subset of units.

In Pajek the 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

A partition of the set of units U is formed by clusters.

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

if:

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

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

their gender (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 + 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 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.

We are also interested in the subnetwork (vertices and lines) which the vertices in a selected cluster induce.

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

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

We will extract a selected cluster which is determined by a partition even more often. 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 input clusters which we want to extract.

Components

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

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

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

Example: Let the vertices of network correspond to buildings in the city, and lines 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, belong 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, belong to the same weakly connected component (one way streets are no restriction).

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

- 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 the network is undirected, a strongly connected component os the same as a weakly connected one, so they are simply called connected components.

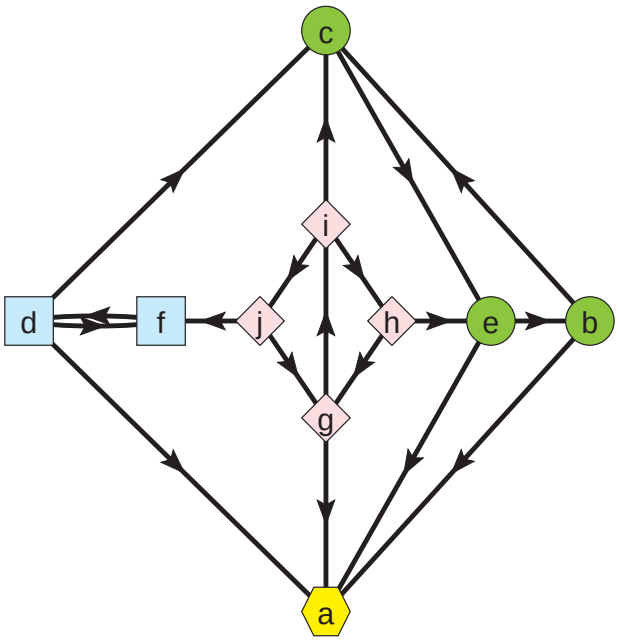
Connection between ***walks*** in a network and ***components***

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.

A network is called ***strongly/weakly connected*** if every pair of vertices is strongly/weakly connected.

Example



The 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.

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, house, man, woman or empty:

```
*Vertices 8
1 "a" box
2 "b" ellipse
3 "c" diamond
4 "d" triangle
5 "e" house
6 "f" man
7 "g" woman
8 "h" empty
..............
```

House, man, and woman are displayed only when exported to EPS or SVG. In Draw window house is replaced by a triangle, man by a box, and woman by an ellipse.

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

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

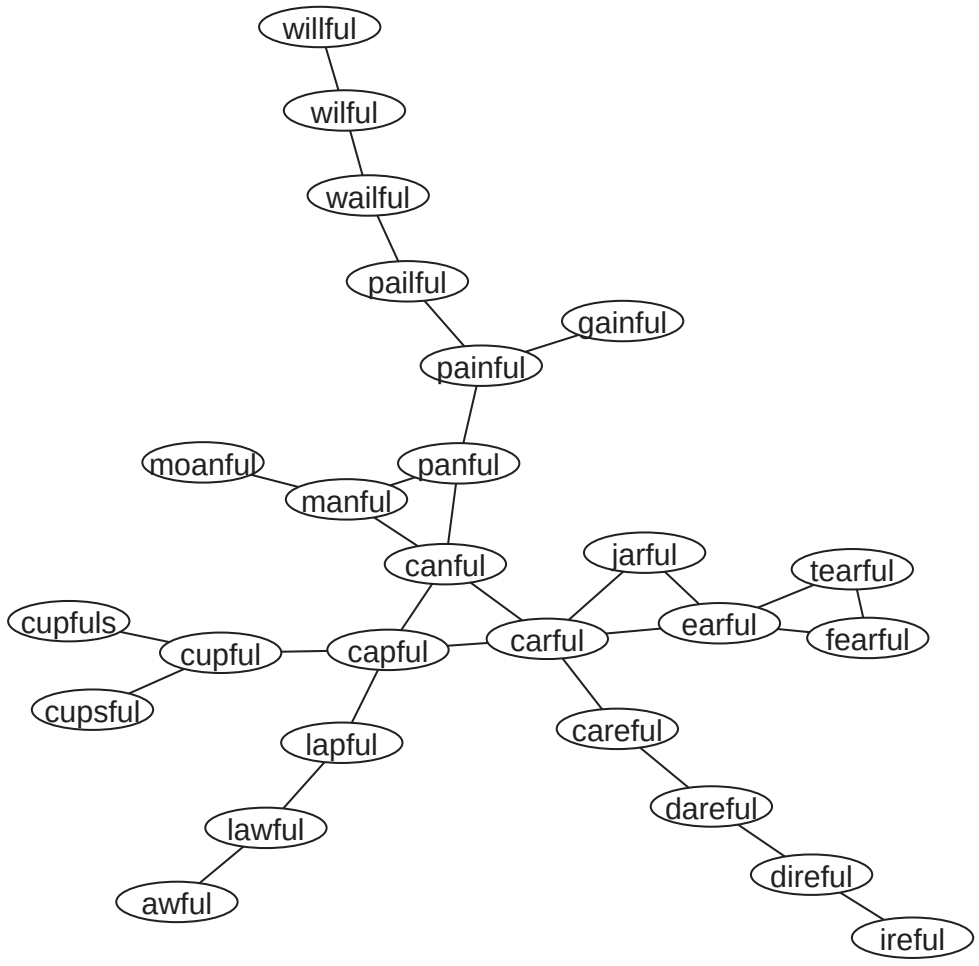
What is the number of components? 17903

Number of vertices in the largest component? 24831

Size of the smallest component? 1 (isolated vertex)

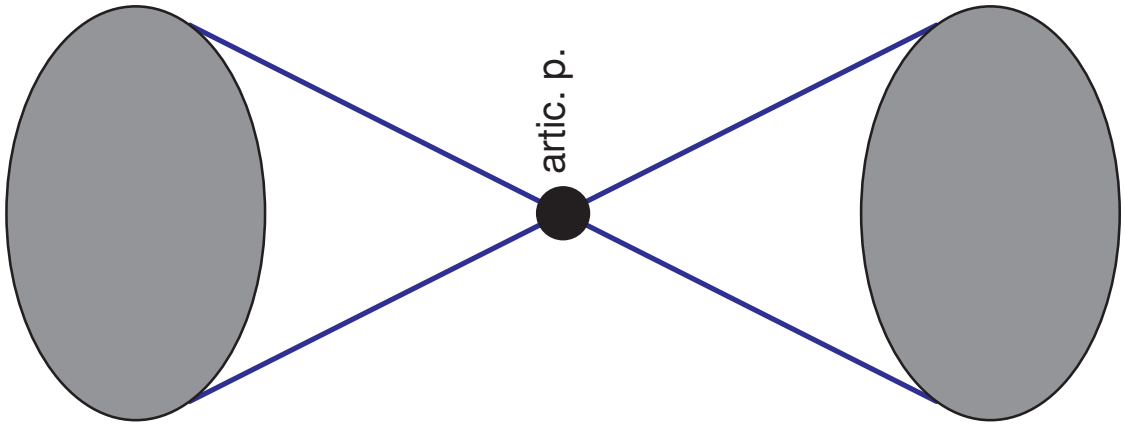
Extract and draw one of the smallest components.

One of the smallest components:



Biconnected components

Lets take the undirected connected network. Vertex a of the network is ***an 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.

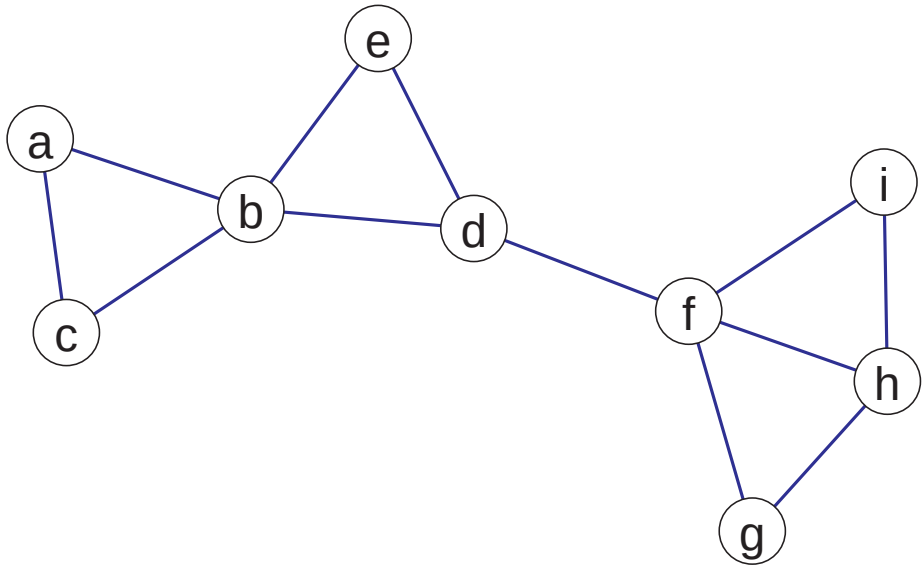


A network is called *biconnected* if for every triple of vertices a, v and w there exists a chain between v and w which does not include vertex a.

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

A biconnected network does not contain any articulation point.

Example

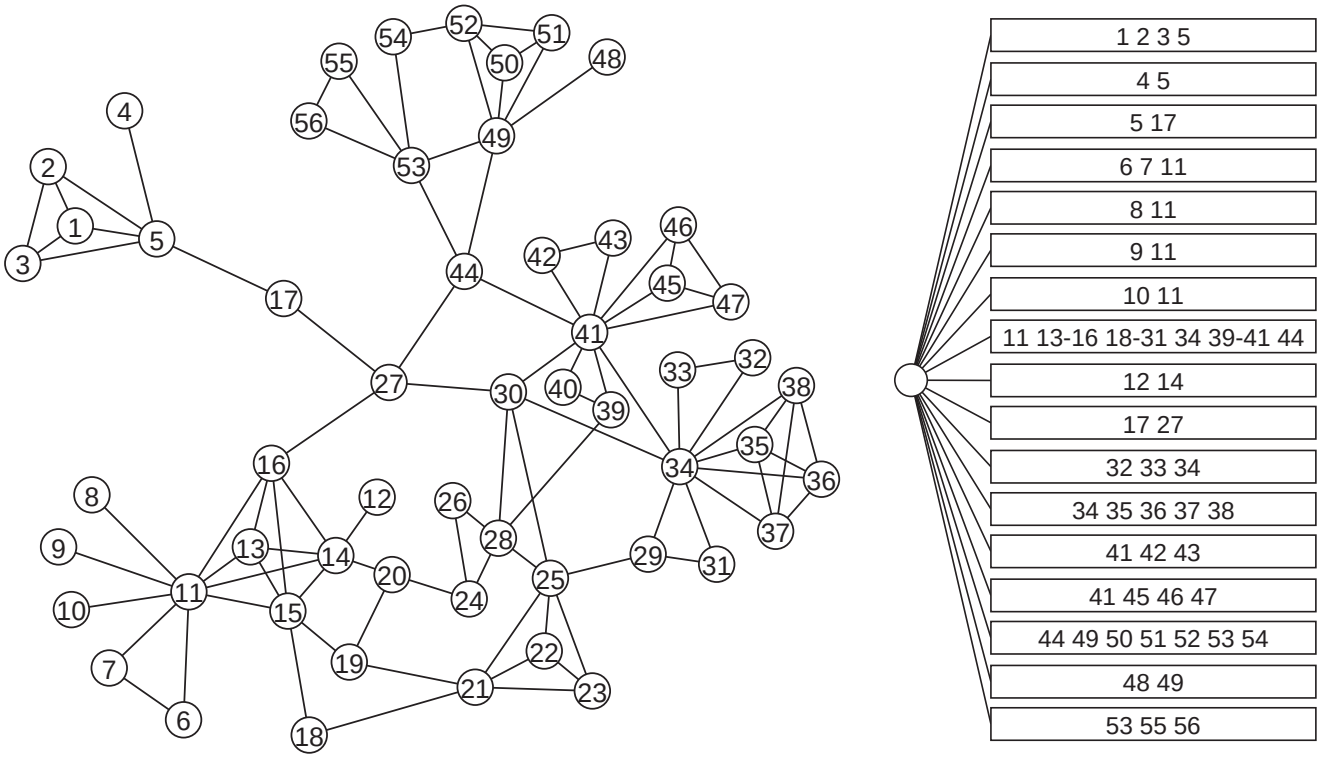


Te network in the picture (aho1.net) consists of 4 biconnected components:

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

The articulation points are b, d and f.

A network that consists of 17 biconnected components:



Definition of components using walks

For ***strongly connected*** components the ***direction of lines is important***:
A subset of vertices is called a 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***:
A subset of vertices is called a weakly connected component if there exists *at least one chain* from any vertex to any other vertex from the subset.

A subset of vertices is called a biconnected component if there exist *at least two chains* from any vertex to any other vertex from the subset, with no intermediate vertices in common.

In some occasions it is beneficial that a 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: 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 vector with articulation points (vertices in cluster 1 are not articulation points, all others are). The number tells, into how many pieces a network will fall, when removing a given

articulation point from the network.

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 8 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
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

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.

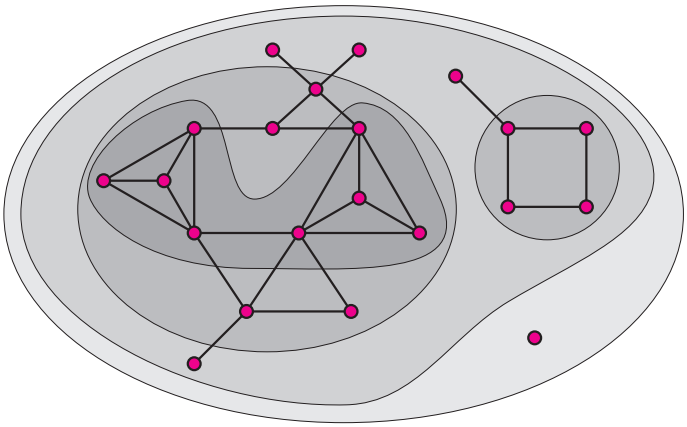
Formally:
Let $N = (V, L)$, $L \subseteq V \times V$ be a network.

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:
A subset of vertices is called ***a 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
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 which clusters to extract.

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

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`.

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?