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.

Paths in a graph

In a *directed graph*:

- A sequence of vertices $(v_1, v_2, \ldots, v_k)$ is called ***a walk*** if

  $$(v_i, v_{i+1}) \in A, i = 1 \ldots k-1$$

  (consequent vertices must be connected with arcs).

- A sequence of vertices $(v_1, v_2, \ldots, v_k)$ is called ***a chain*** if

  $$(v_i, v_{i+1}) \in A \text{ or } (v_{i+1}, v_i) \in A, i = 1 \ldots k-1$$

  (consequent vertices must be connected, direction of lines is not important).

- A walk is ***elementary*** if all its vertices, except maybe initial and terminal, are different. We will call an elementary walk *a path*.

- A walk is ***simple*** if all its lines are different.

- If $v_1 = v_k$, the walk is called ***a closed walk*** or ***circuit*** (the walk starts and ends in the same vertex).

- ***A cycle*** is a closed walk of at least three vertices in which all lines are distinct and all vertices except the initial and terminal are distinct.

- The chain starting and ending in the same vertex is called ***a closed chain***.
- If $v_1 = v_k$ and $k = 1$, the circuit is called ***a loop*** (a connection of a vertex to itself).
- ***The length of a path*** is $k - 1$ (the number of lines traveled).
- There can exist several paths between two vertices. The most interesting is ***the shortest path*** (all shortest paths).

  If a relation means *telling news* the shortest path will tell the shortest route of information traveling between two persons. – e. g., if all shortest paths from people to a selected person are short the selected person is well or quickly informed.

  In the case of a relation *was in contact to*, the shortest path tells the most probable way of passing an infection. In this case shortest paths are not welcome.
- The length of the longest shortest path in a graph is called its ***diameter***.

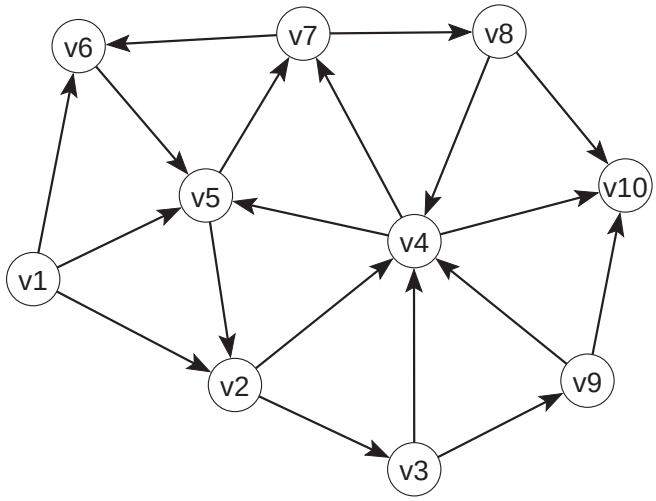
Some interesting results

Networks with large number of vertices usually have short shortest paths among vertices. For example:

- The average length of shortest path of the WWW, with over 800 million vertices, is around 19. Albert, R., Jeong, H., and Barabasi, A.-L. (1999): Diameter of the World-Wide Web. *Nature*, **401**, 130-131. https://www.nature.com/articles/43601
- Social networks (whom do you know) with over six billion individuals are believed to have a average length of shortest path around six. Milgram, S. (1967): The small-world problem. *Psychol. Today*, **2**, 60-67.

  Small world experiment: A psychologist Stanley Milgram made the following experiment with letters: The letter should reach a target person. The persons involved in experiment were asked to send the letter with these instructions to the target person (if they personally know him/her) or (if they do not know him/her personally) to their friend who was more likely to know the target. Letters were sent from Omaha (Nebraska) to target person in Boston (Massachusetts). The average length of the successful paths was 6.

  See: en.wikipedia.org/wiki/Small-world_experiment



$v1 - v6 - v4 - v8$ is not even a chain

$v1 - v6 - v7 - v8$ is a chain, but not a walk

$v8 - v4 - v5 - v2 - v4 - v10$ is a simple, not elementary walk

$v8 - v4 - v5 - v2 - v3 - v9$ is an elementary walk (path)

$v4 - v10 - v8 - v4$ is a closed chain

$v6 - v5 - v7 - v6$ is a cycle

$v1 - v2 - v3 - v9 - v10$ is a path between $v1$ and $v10$, but not a shortest path (length 4).

The shortest path is: $v1 - v2 - v4 - v10$ (length 3).

The diameter is 5: $v7 - v6 - v5 - v2 - v3 - v9$

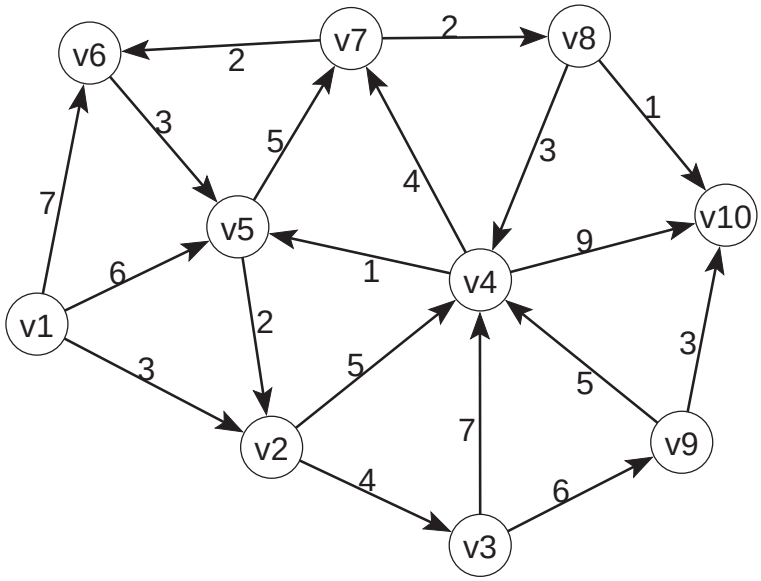
The value of a path between two vertices is the sum of the values of lines which construct the path. If all values of lines are 1, the value of the path is equal to the length of the path.

The shortest path can be defined according to ***the length*** of path or ***the value*** of path.

Example: searching for the shortest path between two cities by known airlines connections:

1. searching for the path so that *number of flights* is the lowest (shortest path according to length of path).
2. searching for the path so that the
   - *total time* spent in planes is the lowest (values of lines are duration of flights)
   - *total cost* is the lowest (values of lines are costs of flights)
   - *total length of the path* is the lowest (values of lines are distances between airports)

Take the same example as before, but let the lines have the following values (network flow2.net):



The path with the lowest value between $v1$ and $v10$ is:
$\boxed{v1 - v5 - v7 - v8 - v10}$ (length is 4, value is 14).

The shortest path according to number of lines between $v1$ and $v10$ is:
$\boxed{v1 - v2 - v4 - v10}$ (length is 3, value is 17).

Optimal solutions according to both criteria are very different in this case.

The chain with the lowest value between $v1$ and $v10$ is:
$\boxed{v1 - v2 - v5 - v4 - v8 - v10}$ (length is 5, value is 10).

Shortest paths in program Pajek

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 (network from last picture):
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 answer 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, . . .

For very small networks you can compute matrix of distances (according to length) among all vertices, and how many shortest paths are connecting vertices using Network/Create New Network/SubNetwork with Paths/ Geodesics Matrices*

Shortest path length matrix (flow2.net):

	1	2	3	4	5	6	7	8	9	10
v1	?	1	2	2	1	1	2	3	3	3
v2	?	3	1	1	2	3	2	3	2	2
v3	?	3	4	1	2	3	2	3	1	2
v4	?	2	3	3	1	2	1	2	4	1
v5	?	1	2	2	3	2	1	2	3	3
v6	?	2	3	3	1	3	2	3	4	4
v7	?	3	4	2	2	1	3	1	5	2
v8	?	3	4	1	2	3	2	3	5	1
v9	?	3	4	1	2	3	2	3	5	1
v10	?	?	?	?	?	?	?	?	?	?

k-neighbours

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

In the previous example, 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 **Maximum 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 (999999998).

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. When asked
Select minimum frequency leave value 1 (in frequency distribution the class is shown if there is at least one unit in it). When asked **Display Highest+/Lowest- or Interval of Values** input -20. If we want to display 20 the most distant

vertices we input 20.

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] (enter 0-2). 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 Network,
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 are available in Partition. . . .

Acyclic network

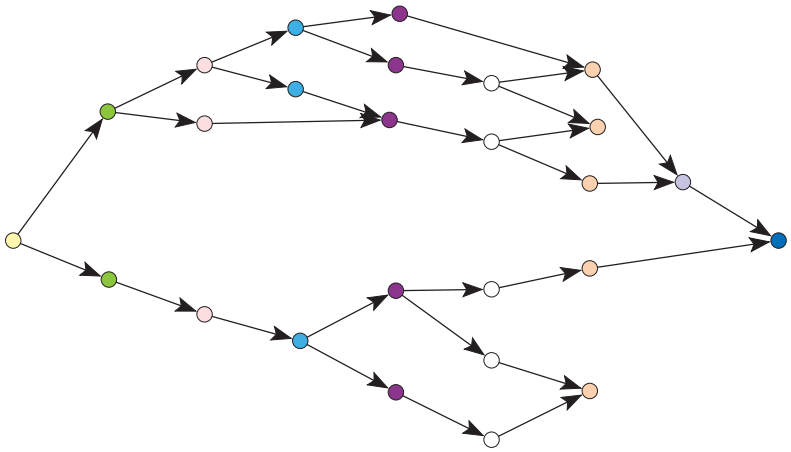
An acyclic network is a special kind of directed network:

A directed network is *acyclic* if it contains no cycles.

If we start a path from any vertex of a network and follow the directions of lines there is no way to return to the initial vertex.

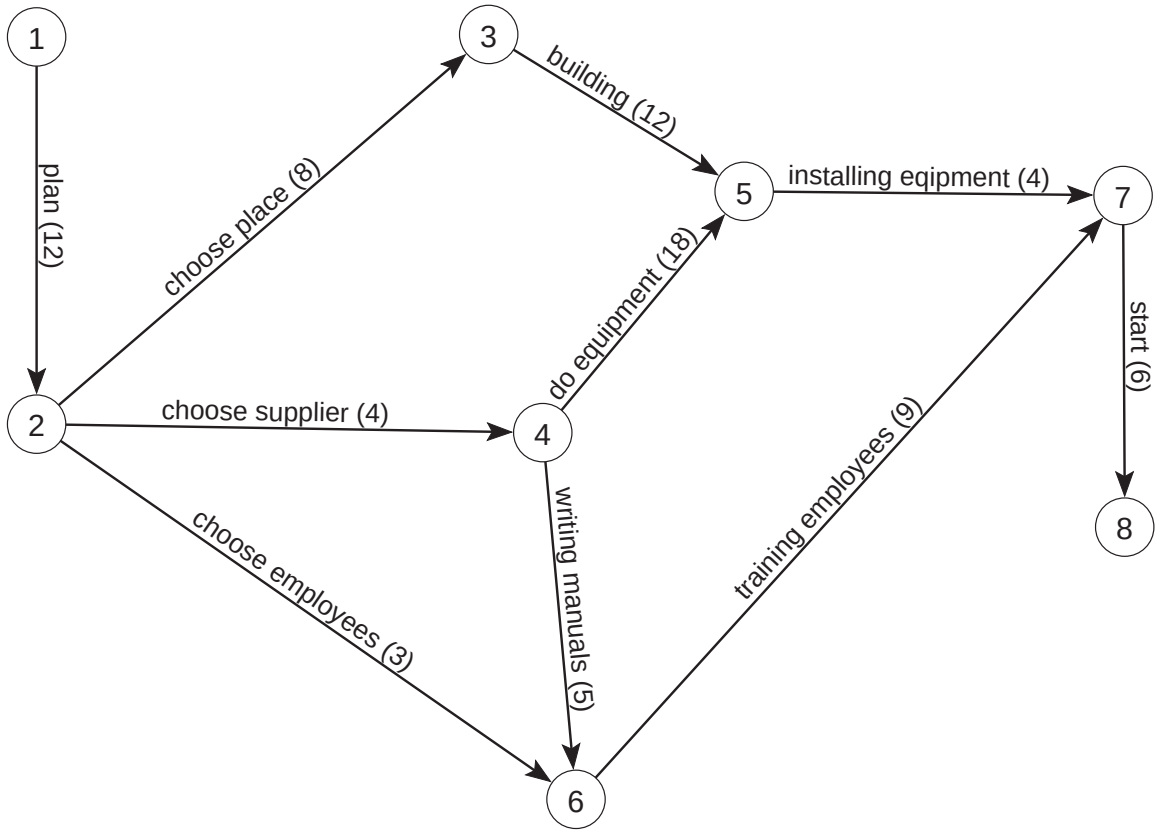
Examples of acyclic networks:

Citation networks



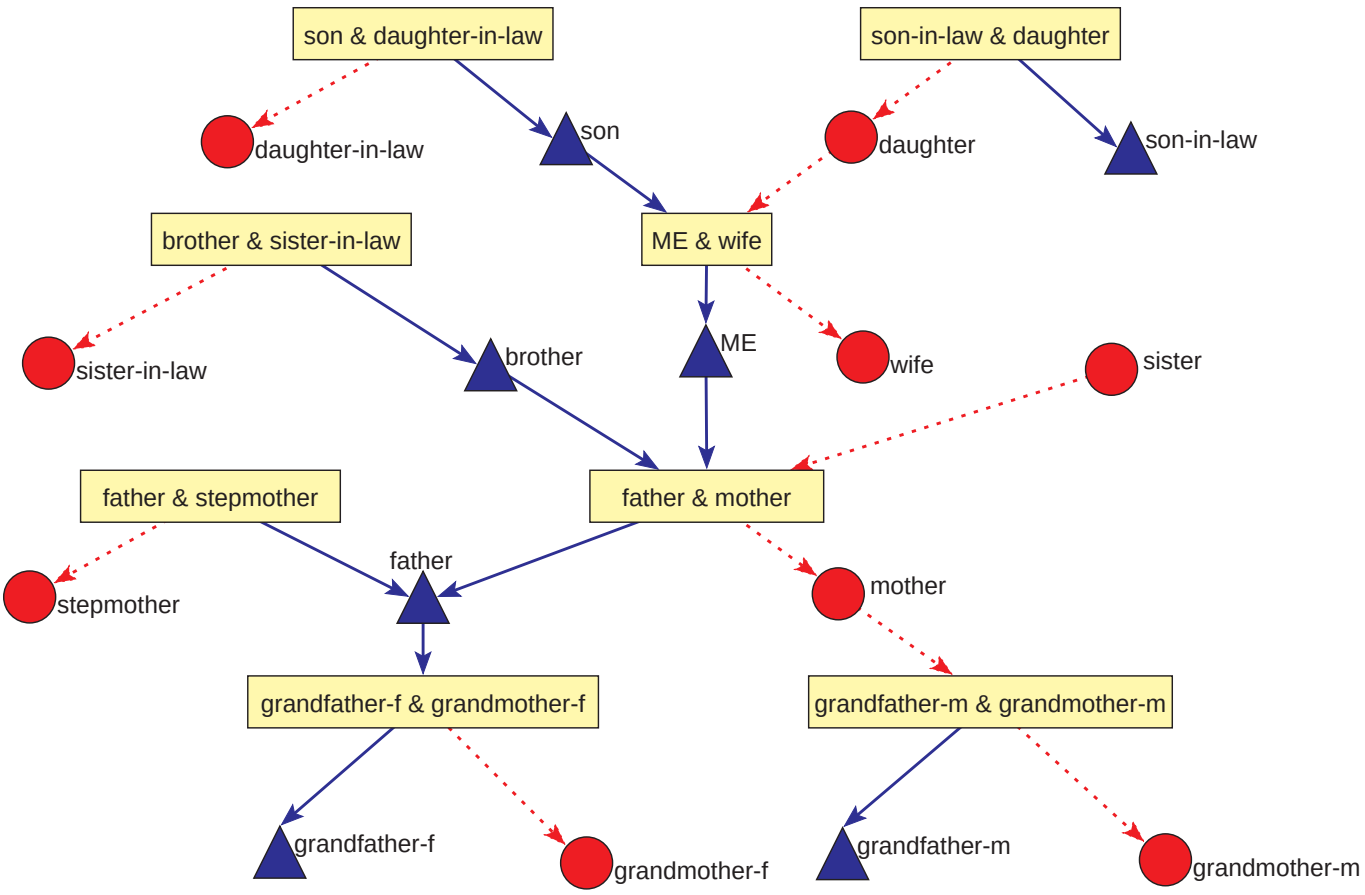
In a citation network it is interesting to find out which author is the most influential and which citation is the most crucial. Both measures are built by counting all paths passing through vertices / lines (the number of paths in a finite acyclic network is finite).

CPM Network (*Critical Path Method*).



In a CPM network we want to find *a critical path* which determines the time needed to finish the whole project.
In our example the critical path is 1 – 2 – 4 – 5 – 7 – 8, and the duration is 44.

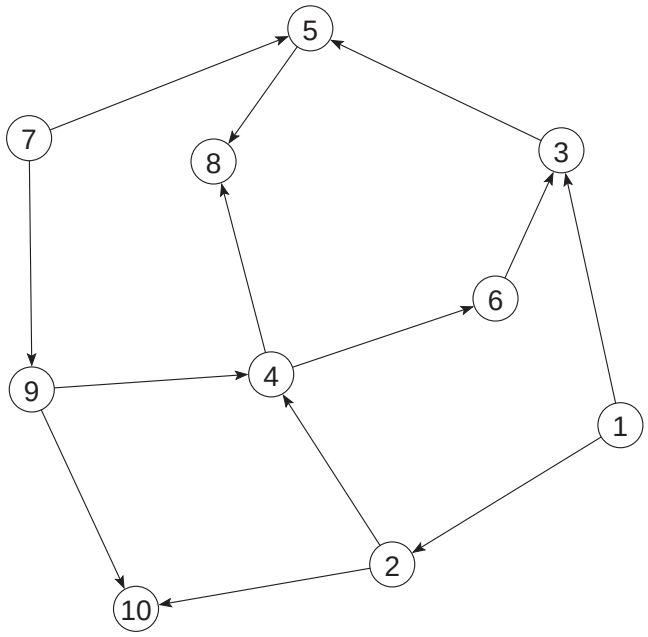
Genealogies



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.

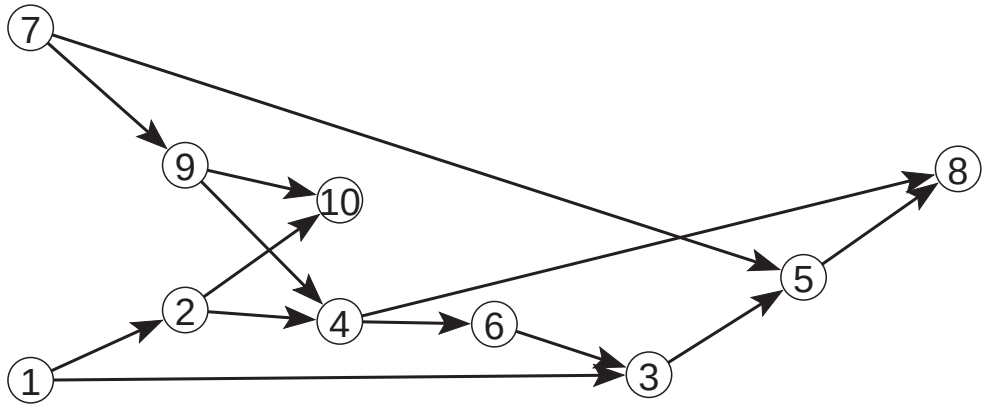
Example: Vertices in the acyclic network



have the following **depths**

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

Depths can be used for drawing the acyclic network in **layers**. If we use layers in x direction we get:



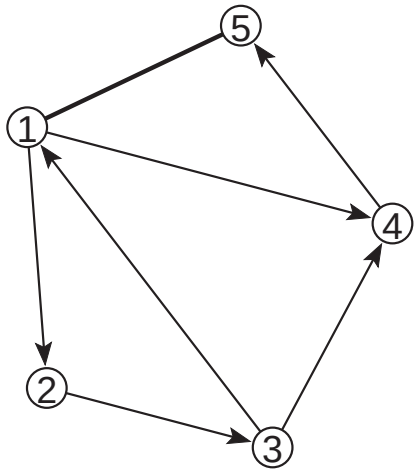
In Pajek depths are computed using
Network/Acyclic Network/Depth Partition/Acyclic

If the command Draw/Network+First 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.

Degree of a vertex

Input degree: number of incoming lines.
Output degree: number of outgoing lines.
All degree: total number of incoming and outgoing lines.



For the graph in the picture we have:
input degrees: (2, 1, 1, 2, 2)
output degrees: (3, 1, 2, 1, 1)
all degrees: (4, 2, 3, 3, 2)

In Pajek degrees are computed using
Network/Create Partition/Degree
and selecting Input, Output or All. For smaller networks the result can be displayed by double clicking the partition window, selecting File/Partition/Edit or selecting the corresponding

icon.

After computing degrees we can show degrees of vertices using different colors by selecting Draw/Network+First Partition. If we transform the partition into a vector by Partition/Copy to Vector we can draw a picture of the network using Draw/Network+First Vector where the sizes of vertices are proportional to their degrees. Both properties (color and size) are used for drawing if the command Draw/Network+First Partition+First Vector is used.

In the case of larger networks we are mostly interested only in the vertices having the highest degrees. We are interested in the frequency distribution of degrees, too. These two results can be obtained using Partition/Info. Details are explained in the definition of k-neighbours. The difference is that in the case of k-neighbours we were interested in the closest vertices and we input negative number and here we are interested in vertices having the highest degrees and we must input positive number.

Example: Find words having the highest number of neighbours in (dic28.net).

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 the 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.

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 according to the length and value of paths.
   Find shortest chains according to both criteria too.

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).

5. Draw an acyclic network from wirth.net in layers with as few crossings as possible.

6. Find words with the highest degree in dic28.net.