Nice pictures of networks

Some properties of nice pictures of networks:

- not too many crossing of lines,
  graph that can be drawn without crossing of lines is called *planar* graph)
- not too small angles among lines that have one vertex in common
- not too long, not 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. Decimal delimiter is point. 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.

Other commands in Options/Transform are: (commands can be applied to the whole picture or just selected part of it):

Resize – extend/shrink the picture for some factor in selected directions

Translate – translate the picture in selected direction

Rotate 2D – rotate the picture in xy plane for selected angle

---

The properties of nice picture can be checked in Pajek in the following way:

Select the desired property using Info in Draw window, press Ctrl+I (or select Info/Layout in main window).

- *Closest Vertices* – find the closest vertices, and color them yellow. In another window the distance among vertices is written.
- *Smallest Angle* – select the smallest angle among lines having one vertex in common. The corresponding three vertices are colored green. The angle is written in another window.
- *Shortest/Longest Line* – find the shortest line (color the corresponding two vertices red) and longest line (color the corresponding two vertices blue). The length of the two lines is displayed.
- *No. of Crossings* – count the number of crossings of lines (corresponding vertices are colored pink).
- *Vertex Closest to Line* – find the vertex that is the closest to any line. Corresponding three vertices are colored

white. The distance of vertex from line is also displayed.

Example of large network: connections among words in dictionary

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

- changing single character (e. g. work – word)
- adding / removing 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 89.038 edges. Network is sparse: density is 0.0000642.

File: dic28.net.

Paths in a graph

In *directed graph*:

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

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

  (consequent vertices must be connected with arcs).

- Sequence of vertices $(v_1, v_2, \ldots, v_k)$ is called ***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).

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

- Walk is ***simple***, if all its lines are different.

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

- ***Cycle*** is closed walk of at least three nodes in which all lines are distinct and all vertices except initial and terminal are distinct.

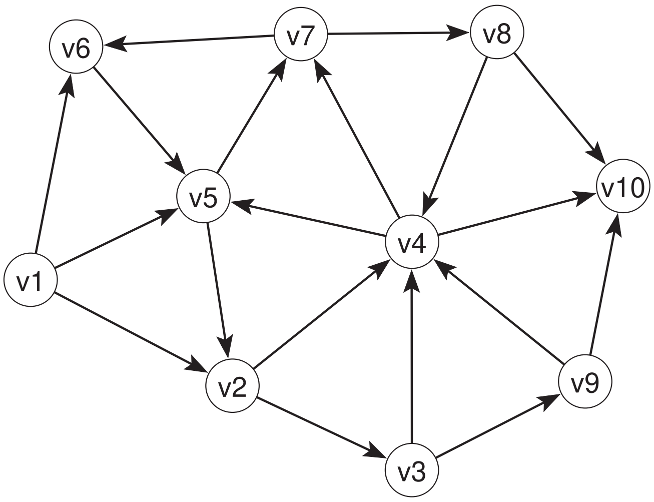
- Chain starting and ending in the same vertex is called

closed chain.

- If $v_1 = v_k$ and $k = 1$, the circuit is called ***loop*** (connection of vertex to itself).
- ***Length of path*** is $k - 1$ (number of lines traveled).
- There can exist several paths among two vertices. The most interesting is ***the shortest path*** (all shortest paths).

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

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



$\boxed{v1 - v6 - v4 - v8}$ is not even a chain

$\boxed{v1 - v6 - v7 - v8}$ is a chain, it is not a walk

$\boxed{v8 - v4 - v5 - v2 - v4 - v10}$ is simple, not elementary walk

$\boxed{v8 - v4 - v5 - v2 - v3 - v9}$ is elementary walk (path)

$\boxed{v4 - v10 - v8 - v4}$ is closed chain

$\boxed{v6 - v5 - v7 - v6}$ is cycle

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

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

Diameter is 5: $\boxed{v7 - v6 - v5 - v2 - v3 - v9}$

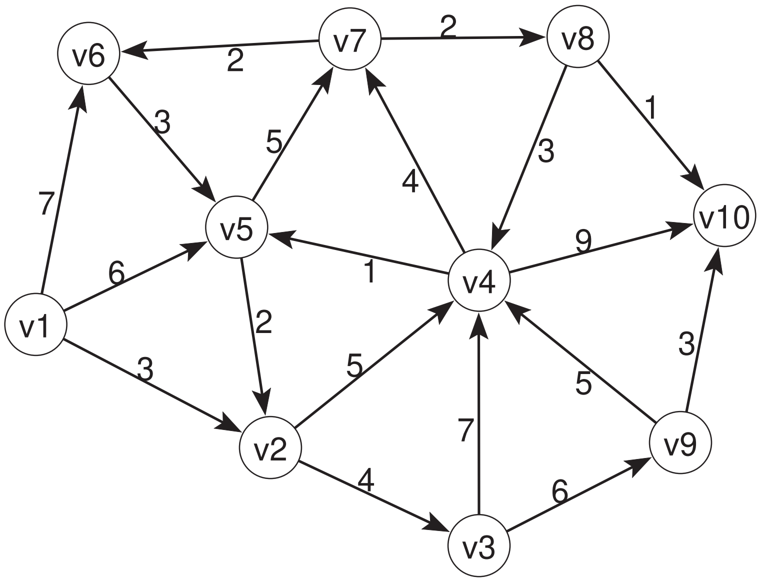
Value of path between two vertices is the sum of values of lines, which construct the path. If all values of lines are 1, the value of path is equal to the length of path.

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

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

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
   - *total time* of seating 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 path* is the lowest (values of lines are distances between airports)

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



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.

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
Net/Paths between 2 vertices/One Shortest
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 selected two vertices, vertices that lie on the shortest path and corresponding lines. The resulting path can be drawn using
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. On the other hand most operations in Pajek returns as a result another object (new network, partition...) which can be further analysed.

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

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

We can find **chains** (direction of lines are not important) in directed network so that we first transform directed lines to undirected using Net/Transform/Arcs to Edges/All.

Example: flow2.net (network from last picture):
find the shortest path between $v1$ and $v10$ according to both criteria
find the shortest 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
Net/Paths between 2 vertices/All Shortest
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 – path does not exist.

Diameter of the network can be found using
Net/Paths between 2 vertices/Diameter
Pajek returns only the two vertices that are the most far 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 very time consuming operation – it can be performed on smaller networks only.

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

white – yellow, engaged – skeptics, ...

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 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: Net/k-Neighbours/Output
We input the selected vertex (number or label) and when asked **Maximal distance** input 0 – we want to check all distances.

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

Distances of vertices that cannot be reached from selected vertex are set to some large number (999998).

Therefore: using Net/k-Neighbours/Output we compute in how many steps we can reach all other vertices from selected vertex.

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

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: Info/Partition and when asked
Number of vertices with highest / lowest value
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/Extract from Network/Partition
and input for the lower limit $\boxed{0}$ and for the upper $\boxed{2}$.

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 Nets,
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. . . .

Acyclic network

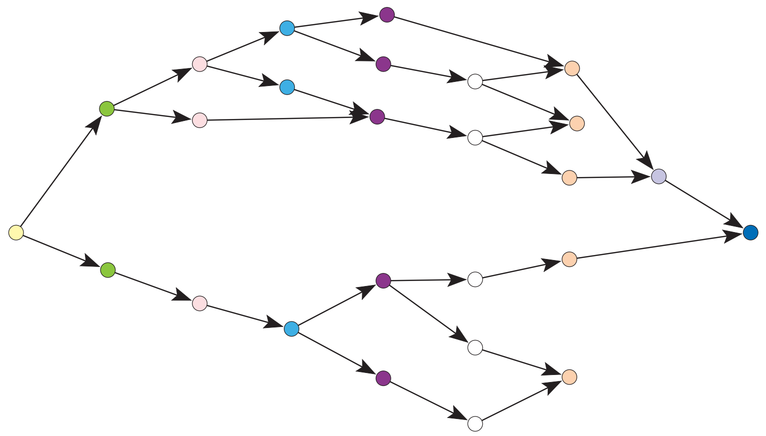
Acyclic network is a special kind of directed network:

Directed network is *acyclic*, if there are no cycles in it.

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

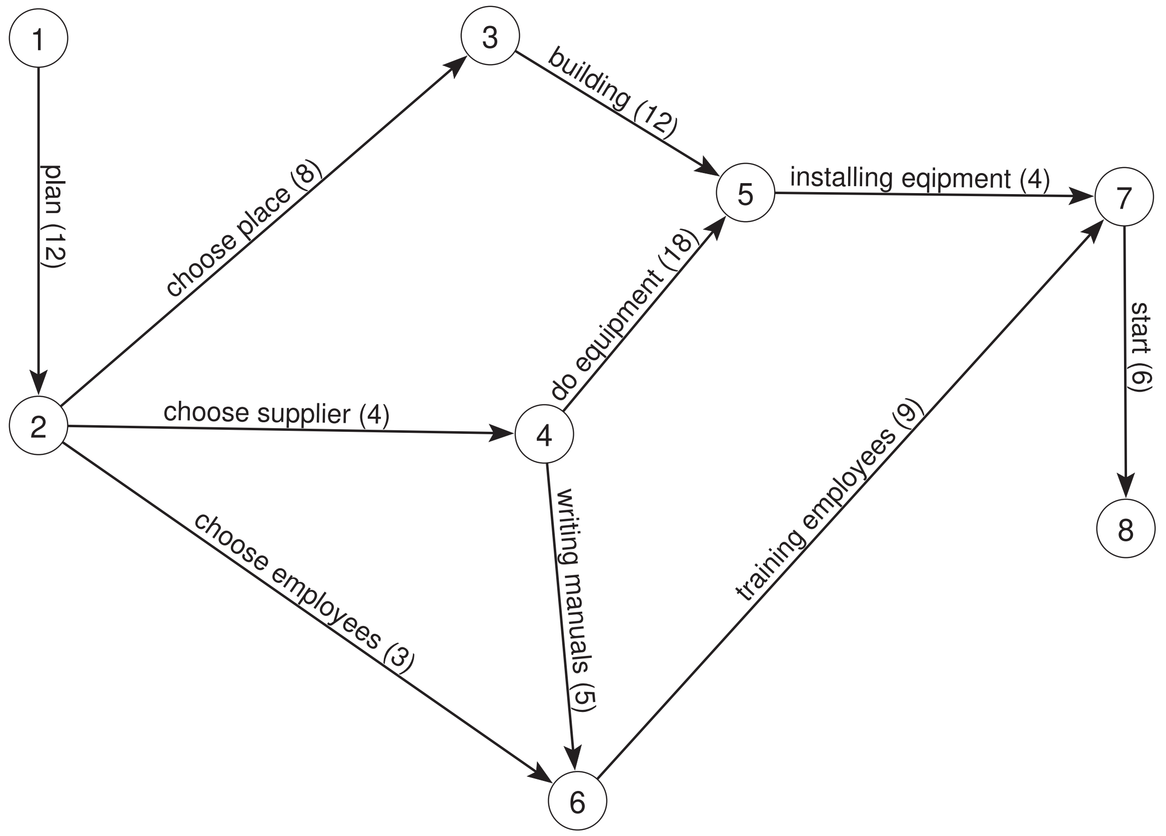
<u>**Examples of acyclic networks:**</u>

Citation networks



In 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 on counting all paths passing through vertices / lines (number of paths in finite acyclic network is finite).

CPM Network (*Critical Path Method*).

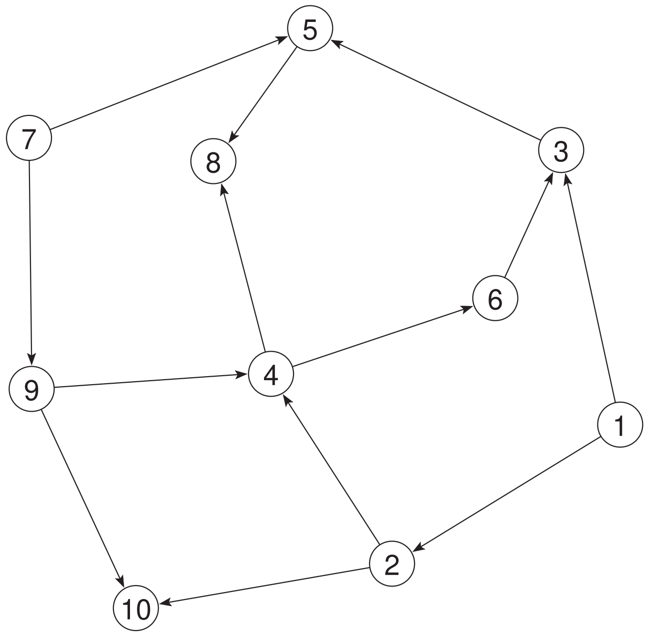


In CPM network we want to find *critical path* which determines the time needed to finish the whole project.
In our example the critical path is 1 – 2 – 4 – 5 – 7 – 8, duration is 44.

In 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 acyclic network the ***depth*** can be computed: We assign depth 1 to all first vertices and remove first vertices and corresponding lines out of the network. 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 out of the network and continue the procedure.

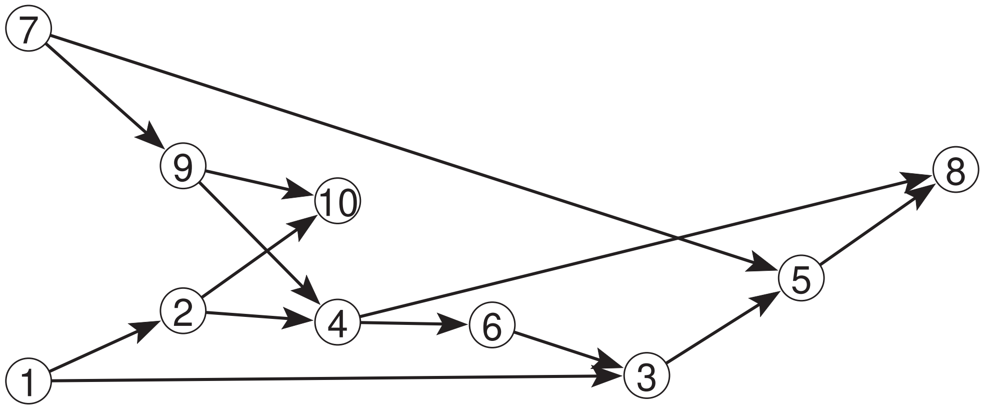
Example: Vertices in acyclic network



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
Net/Partitions/Depth/Acyclic

If command Draw/Draw-Partition is selected afterwards, each vertex is colored using color that corresponds to its depth. Colors used are shown in
http://vlado.fmf.uni-lj.si/pub/networks/pajek/part_col.pdf

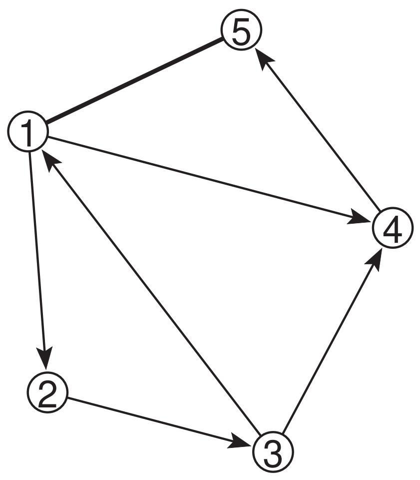
Layout in layers is obtained using Layers/in y direction. If we want to improve the picture by hand, but we want to keep the layers, we can define y coordinate as fixed using
Move/Fix y
y coordinate of any vertex cannot be changed afterwards.

Degree of vertex

Input degree: number of lines coming into vertex.

Output degree: number of lines going out of vertex.

All degree: total number of lines, coming in or going out of vertex.



For the graph in the picture we have:

input degrees: (2, 1, 1, 2, 2)

output degrees: (3, 1, 2, 1, 1)

all degrees: (5, 2, 3, 3, 3)

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

After computing degrees we can show degrees of vertices using different colors by selecting Draw/Draw-Partition. If we transform partition into vector by Partition/Make Vector we can draw picture of network where sizes of vertices are proportional to their degrees using Draw/Draw-Vector. Both properties (color and size) are used for drawing if command Draw/Draw-Partition-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 frequency distribution of degrees, too. These two results can be obtained using Info/Partition. Details are explained in definition of k-neighbours. The difference is that we were interested in the closest vertices in the case of k-neighbours (we input negative number), here we are interested in vertices having the highest degrees (we must input positive number).

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

Transforming two-mode networks to ordinary networks in Pajek

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

```
*Vertices 32 18
 1 "EVELYN"
 2 "LAURA"
 3 "THERESA"
 4 "BRENDA"
 5 ČHARLOTTE"
 6 "FRANCES"
 7 "ELEANOR"
 8 "PEARL"
 9 "RUTH"
10 "VERNE"
11 "MYRNA"
12 "KATHERINE"
13 ŠYLVIA"
14 "NORA"
15 "HELEN"
16 "DOROTHY"
17 "OLIVIA"
18 "FLORA"
19 "E1"
20 "E2"
21 "E3"
22 "E4"
23 "E5"
24 "E6"
25 "E7"
26 "E8"
27 "E9"
28 "E10"
29 "E11"
30 "E12"
31 "E13"
32 "E14"
*Edgeslist
1 19 20 21 22 23 24 26 27
2 19 20 21 23 24 25 26
3 20 21 22 23 24 25 26 27
4 19 21 22 23 24 25 26
5 21 22 23 25
6 21 23 24 26
7 23 24 25 26
8 24 26 27
9 23 25 26 27
10 25 26 27 30
11 26 27 28 30
12 26 27 28 30 31 32
13 25 26 27 28 30 31 32
14 24 25 27 28 29 30 31 32
15 25 26 28 29 30 31 32
16 26 27 28 30
17 27 29
18 27 29
```

The only difference according to ordinary networks is that we have to give the total number of vertices (in our case 32) and number of vertices that belong to first subset (in our case 18 women) in the *Vertices statement.

Network is transformed into ordinary network, where the vertices are elements from the first subset (in our case women) using Net/Transform/2-Mode to 1-Mode/Rows.

If we want to get network with elements from the second subset we use Net/Transform/2-Mode to 1-Mode/Columns.

The result is a network with values on lines in both cases. If we want to keep only the lines having enough large values and delete the others, we use
Net/Transform/Remove/lines with value/lower than
and input the limit.

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

Example: net.net.

If we have network with values of lines we can use Options/Values of Lines to determine if values of lines are similarities, dissimilarities, or we can forget values of lines. 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 the layout is optimized when using energy algorithms.

Using <u>Options/ScrollBar On/Off</u> scrollbar is set into <u>Draw</u> window. Scrollbar works in two different ways:

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

When exporting layout in EPS format several parameters can be set in <u>Export/Options</u>. In this way we can define in details how the picture should look like. EPS pictures can be displayed using program <u>GsView</u>.

Examples

1. Find shortest paths between $v1$ and $v10$ in flow2.net according to length and value of path.
   Find shortest chains according to both criteria too.
2. In 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 distances of all words from yellow. Extract and draw subnetwork including all words that are on distance at most 3 from yellow (including yellow too).
5. Draw acyclic network wirth.net in layers with as few crossings as possible.
6. Find words with the highest degree in dic28.net.