

# User Manual:

## Stress Minimization Layouter for VANTED

—A VANTED Add-On for Layouting Large Graphs with a Stress Minimization Algorithm—

<b>Introduction</b>	<b>2</b>
Overview	2
Libraries	2
Copyright	2
Acronyms and Abbreviations	2
<b>Preconditions and Behaviour</b>	<b>3</b>
<b>Work with Stress Minimization</b>	<b>3</b>
Provided Functions	3
Limitations	4
General Options	5
Stop criteria	6
Weights	6
Initial layout	6
Iterative algorithm	6
General layout	6
Edge scaling factor	6
Edge length minimum	7
Remove edge bends	7
Iterations undoable	7
Animate iterations	7
Move into view	7
Parallelism (advanced)	8
Initial layouters	8
PivotMDS	9
Do nothing	9
Iterative algorithms	9
Intuitive Algorithm	10
Exemplary Workflow	11
<b>Comparison to ‘Force Directed’</b>	<b>14</b>

# Introduction

## Overview

This add-on provides a layout algorithm based on stress minimization. Stress minimization is a technique that tries to adjust the graph theoretical distance between two nodes to the actual Euclidean distance between the nodes. It is designed to layout graphs up to 5000 nodes, but larger graphs can also be layouted with longer running times. Compared to 'Force Directed' implemented in *VANTED*, it is much faster. In most cases the overall layout is better (disregarding execution time) because local minima occur less often. To gain good performance a good initial layout is needed, therefore multiple pre-layouts are offered. This manual was written for version 0.1 of the add-on.

## Libraries

This add-on only needs some of the libraries already used by *VANTED* and does not require any additional libraries. When installing the add-on it must be ensured that these libraries are also available to the add-on (in most cases *VANTED* is already delivered pre-packaged with these libraries).

## Copyright

This software is licensed under the WTFPL (see <http://www.wtfpl.net/about/>).

## Acronyms and Abbreviations

VANTED — Visualisation and Analysis of Networks conTaining Experimental Data

WTFPL — Do What The Fuck You Want To Public License

SM — Stress Minimization (method)

GUI — Graphical user interface.

# Preconditions and Behaviour

To perform stress minimization, a graph must be loaded in *VANTED* (see “*VANTED* User Manual”). The graph to be layouted must not be empty. The ‘Stress Minimization’ layout cannot be used multiple times on the same graph simultaneously.

The layout will only work on the currently selected nodes. If none are selected, the whole graph will be layouted. Not selected nodes will be ignored.

Furthermore the algorithm will ignore any direction, loops, weights and bends of the edges between the nodes.

After completion the connected components of the graph will be layouted comparable to the existing “Layout unconnected subgraphs on Grid” layout.

## Work with Stress Minimization

### Provided Functions

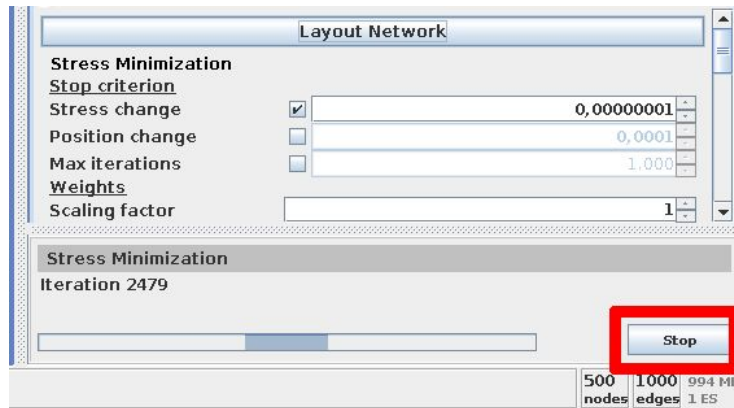
The layouted provided by this plugin can be found under the name “Stress Minimization” in the list of layouts in the “Layout” tab, that appears after a graph has been loaded.

Once the layouter has been selected and the options have been configured the user can start the layout on the graph in the currently selected view. It is not possible to start the algorithm on the same graph again if a stress minimization process is currently running on the graph, but another graph (view) can be opened and the algorithm can be started on this other graph again. Each currently processed graph can have its own options. They will not influence each other.

The provided number fields allow up to 17 decimal places for any decimal numbers. If the next or previous value is too coarse, a number with higher number of decimal places can be entered manually.

The task window opened when the algorithm is started shows the current status of the algorithm. While preprocessing, the currently performed action is shown. The possible number in front on the action is the number of the connected component being processed at the time.

The add-on enables the user to stop the algorithm at every iteration step (if it is run as a background task). It will then be finished and the intermediate result will be used as layout for the graph.



The location of the stop button.

## Limitations

Due to the procedure of 'Stress Minimization', it may come to unfavorable behaviour in some graphs. In the following, some cases are explained with possible strategies to improve the behaviour:

For paths the optimal layout is a straight line. This leads to the problem, that the SM algorithm approximates (seen as very small movements) slowly to the optimum. To prevent this, a maximum number of iterations can be set at the beginning (see [stop criteria options](#)) or the procedure can be aborted manually.

In the case of deep tree graphs, it can be observed that overlaps can quickly occur. This is because in the default settings a low weight is given to nodes with a high distance. To prevent this, the weight function (see [Weights](#)) can be adjusted.

If graphs are larger than 5000 nodes, the running time may be longer. This can be improved by loosening the stop conditions. However, this goes in favor of layout quality.

With regards to the stopping functionality one must keep in mind that in the current version of the add-on, it is only possible to stop the iterations, but not the preprocessing. So if the distance calculation takes very long, it cannot be stopped and the initial layout will always be applied (but can always be undone, too).

The algorithm caches some heavily used but hard to calculate values like distance information. As the distance between every node pair has to be cached, the required memory increases quadratically with the number of nodes in the graph. This can result in VANTED running out of memory for larger graphs (more 10'000 nodes, depending on the machine).

This can be circumvented by increasing the heap and stack size of the JVM or laying out only subgraphs by selecting them.

## General Options

The algorithm provides some options that can be adjusted to get a better layout or increase the running time. Every option provides a short description as a tooltip.

This is the options GUI of the 'stress minimization' layouter

### Stop criteria

Select one or multiple stop criteria for the algorithm. This can be used to influence running time and quality. A criterion can be enabled or disabled by clicking on the checkbox next to the input field. The algorithm will stop if any enabled stop criterion is encountered. It will never stop automatically if all criteria are disabled and must be stopped manually.

#### Stress change

A *decimal* value that determines the change in stress to be desired. If an actual value smaller than this value is reached, the algorithm will stop. A small value will cause the algorithm to make more iterations.

#### Position change

A *decimal* value that determines the change in positions to be desired. If an actual value smaller than this value is reached, the algorithm will stop. A small value will also cause the algorithm to make more iterations.

## Max iterations

An non-negative *integer* value that determines the maximum iterations to perform. The algorithm will stop after the specified amount of iterations is done.

This value can be set to 0 if only the initial layout should be performed.

## Weights

The weight function is used to determine how much the distance between nodes can influence their graph position. A small value will mean that they have a very small influence, a big one that they will have a great influence on each other. This will result in very small or very big movements respectively. This can increase or lower the running time. The weights are calculated by the following function  $\alpha \cdot (\delta_{ij})^\beta$ , where  $\delta_{ij}$  is the distance between the nodes  $i$  and  $j$ ,  $\alpha$  the constant scaling factor and  $\beta$  is the power the distance is raised to.  $\alpha$  and  $\beta$  are both decimal values that are configurable using the options below.

## Scaling factor

The constant, non-negative *decimal* weight scaling factor mentioned as  $\alpha$  above.

## Distance power

The constant *decimal* weight scaling factor mentioned as  $\beta$  above.

## Initial layout

Layout procedure to apply, before the actual SM algorithm gets executed. A good initial layout can improve the quality and running time of the algorithm.

All default layouters are described [below](#). (A short description of the currently selected layouter is available when hovering over the combo box.)

## Iterative algorithm

The algorithm that should be used to calculate new positions in every iteration step. The used iterative algorithm will do the main work in the SM algorithm so it has a great influence on the resulting quality and running time.

All iterative algorithms are described [below](#). (A short description of the currently selected iterative algorithm is available when hovering over the combo box.)

## General layout

This section contains some general settings for the SM layout algorithm.

## Edge scaling factor

A non-negative *decimal* value representing the amount of space that the algorithm tries to ensure between two nodes (length of the edges) as a factor of the size of the largest node encountered. If the size of the largest node should not be used, this value can be set to

zero. In this case the minimum edge length (see below) will always be used. So the edge length can be set to a constant value.

This value may be set to a smaller value if the graph has some very big nodes, to avoid that the edge lengths become too big and the resulting layout being confusing.

### Edge length minimum

A positive *decimal* value that sets the amount of space that the algorithm at least tries to ensure between two nodes. This value will be used if the scaled length (see above) is smaller than it.

### Remove edge bends

Remove edge bends before starting the algorithm. If there are any bends this will always create an extra undo step before executing the actual algorithm.

Highly recommended because the algorithm does not move edge bends.

### Iterations undoable

Makes each iteration step undoable own undo action (only recommended for small iteration sizes because the amount of undoable actions in *VANTED* is very limited).

If this option is enabled animations will be done regardless of the setting of "Animate iterations".

### Animate iterations

If this is enabled, every position update the algorithm makes will be directly transferred to the displayed graph, thus animating the graph.

Because the redrawing and updating takes time this option may impact performance and should thus be disabled on slow machines.

If "Iterations undoable" is enabled the algorithm will always behave like this option is enabled.

### Move into view

This option only takes effect if "Animate iterations is enabled".

The algorithm will layout and reorder the connected components after every iteration and try to move them into view if this option is enabled. If the graph is very high or wide the user may have to zoom out to see the graph.

Because a lot of bounds may be calculated and a lot of nodes are moved the impact on performance can be even bigger than just using animations. Thus this feature should be disabled on slow machines.

### Parallelism (advanced)

In this section the user may set the options for parallelism (multi-threading) used by the algorithm. These options may improve the running time in some configurations but in most cases its advisable to let these options in their default settings.

These options should only be changed by experienced users.

## Run in background

If this option is enabled, a background task will be created every time the SM algorithm is started. This is visible to the user by the task popup box in the lower right corner of the *VANTED* window. Because the algorithm often takes more than a few seconds to layout a graph, this is highly recommended. If this option is disabled, the *VANTED* GUI will freeze until the algorithm has completed. This also means that the user will not be able to stop the algorithm until it has finished execution.

## Use parallelism

This option determines whether multiple threads are used to run the algorithm. In the current version the calculation of the distances and every connected component in an iteration step are parallelized.

Enabling this increases performance overall but it may be disabled on graphs with a very small node count and few connected components, to avoid the overhead of creating the threads for execution.

## Initial layouters

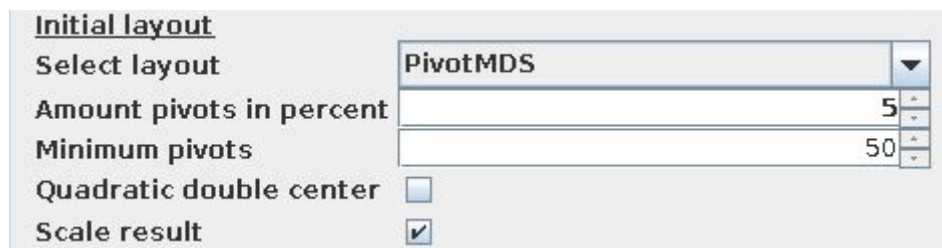
The add-on provides two initial layouters. For a description of what these layouters generally do please refer to the respective [section in General Options](#).

### PivotMDS

A layouter that tries to find nodes that are somewhat evenly distributed in the graph called pivots and calculates a good layout for them. After that the other nodes are positioned relative to these pivots.

This layouter creates a reasonably well layout that can be improved by the SM algorithm. The amount of pivot nodes to be used increases the calculation time and used memory, but will also generally increase the quality of the initial layout. The already existing layout is not regarded by this layouter and will be discarded by it.

'PivotMDS' provides a set of options to further configure its behaviour.



Initial layout	
Select layout	PivotMDS
Amount pivots in percent	5
Minimum pivots	50
Quadratic double center	<input type="checkbox"/>
Scale result	<input checked="" type="checkbox"/>

The options GUI of 'PivotMDS'

### Amount pivots in percent

This *decimal* value determines the amount of pivot nodes that are chosen from the graph. It is interpreted as percent of all nodes. Valid values are between 0 and 100 (including).



If a graph independent, constant value is desired this value should be set to 0. Then the constant value of “Minimum pivots” will be used instead.

### Minimum pivots

This *integer* option sets the minimum amount of nodes that should be used. If the procentual amount of nodes is smaller than this value, this value will be used instead. If this value is bigger than the number of nodes in the graph, the number of nodes in the graph will be used instead.

This value must be bigger or equal to 1. Choosing a very small value for this however is not recommended, because it may result in a very, very bad initial layout of the graph making recovery from it by the SM algorithm almost impossible.

In most cases a value up to 100 will suffice for a good initial layout.

### Quadratic double center

This option determines whether to use the squared distances in the double centering algorithm used in this layouter. The original paper does uses this configuration.

If this option is enabled the graph will scale very big and the SM algorithm may must make many iterations to get it to the desired size. The resulting layout may be better or worse when this option is desired.

If enabled the option “Scale result” should be enabled to reduce this effect.

### Scale result

If this option is enabled, the layout will be scaled down (or up) to a heuristic of the desired size of the graph. Disabling this option may result in layouts with extreme size dimensions, which are harder to recover from for the SM algorithm.

### Do nothing

This layout does not change the current layout of the graph before executing the SM algorithm. It is useful when a rerun of the SM algorithm with for example other stop criteria is desired without losing the old result (thus adding to it).

I can also be used to use a custom initial layout by executing the SM algorithm after the desired “initial layout”, for example the random layout, with this selected as the initial layouter.

The ‘Do nothing’ layouter does not provide any options.

## Iterative algorithms

The add-on provides one iterative algorithm. For a description of what these algorithms generally do please refer to the respective [section in General Options](#).

## Intuitive Algorithm

This algorithm positions the nodes relative to the other nodes with a similar calculation than the stress minimization function, letting each node “vote” for the positions of the other nodes.

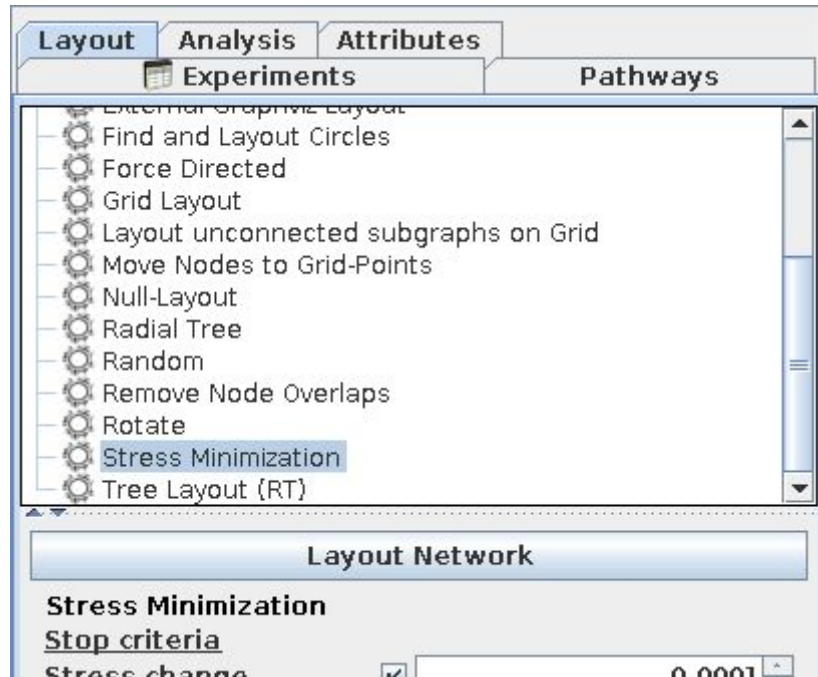
It may happen that the movements are very small. In this case the [distance power](#) can be set to a bigger value to enable more force from more distant nodes.

The “Intuitive algorithm” does not provide any options.

## Exemplary Workflow

The following section describes a more advanced workflow to optimize running time and quality on a tree that does not very well with the default configuration. In the example we will work with an imperfect tree with 4000 nodes and edges generated by *VANTED's* "Watts and Strogatz" generator.

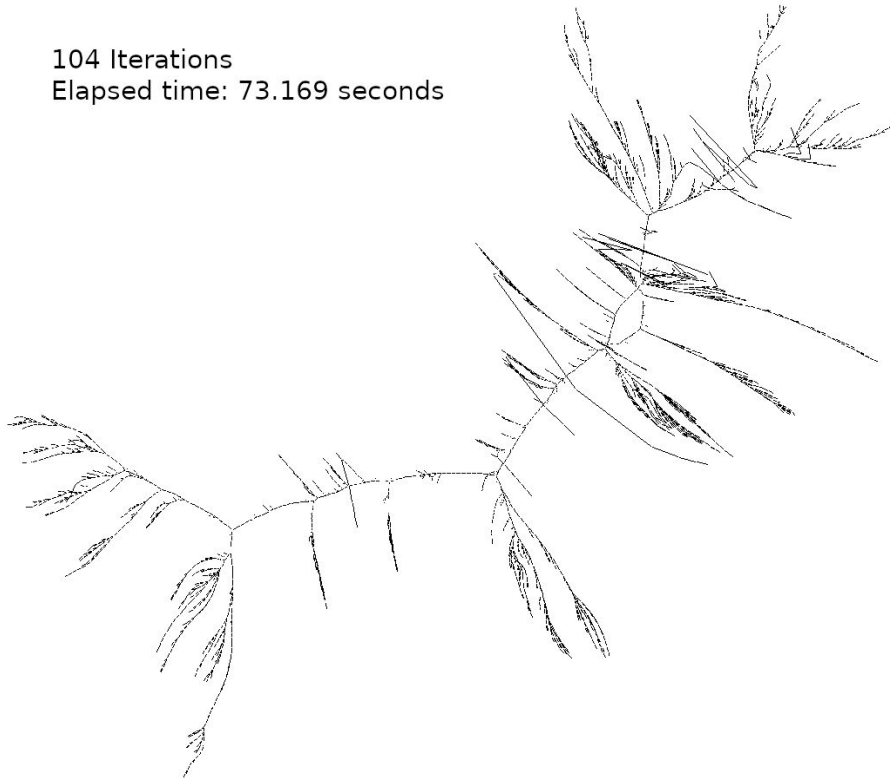
1. Loading the tree in *VANTED* and switching to the layout tab to select the 'Stress Minimization' layout. (If it is unknown how to archive this, the *VANTED* manual may be consulted.)



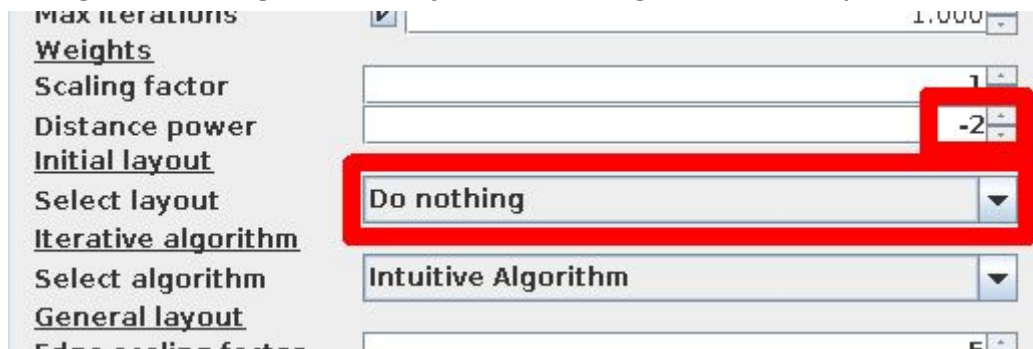
2. Starting the layout process by clicking the "Layout Network" button. The default values should not be changed, especially the options "Distance power" (default "-1") and the "Initial layout" (default "PivotMDS"). The result is shown below. The default power generally results in faster execution times but may produce worse layouts in some circumstances. It is clearly visible that the branches of the tree have not spread out a lot and the tree still looks very cramped.

The result will look similar to this:

104 Iterations  
Elapsed time: 73.169 seconds



3. The next step is going back to the options and changing some of them. We found that setting the "Distance power" in "Weight" to "-2" improves the layout but also takes a lot more time (231 seconds). Instead of recalculating the whole layout we will use the old layout as an initial layout. Selecting "Do Nothing" as Initial layout and setting the "Distance power" to "-2".



4. The last step is layouting the network again. The result should look a lot better and the cumulated time should be smaller than layouting the graph from scratch.

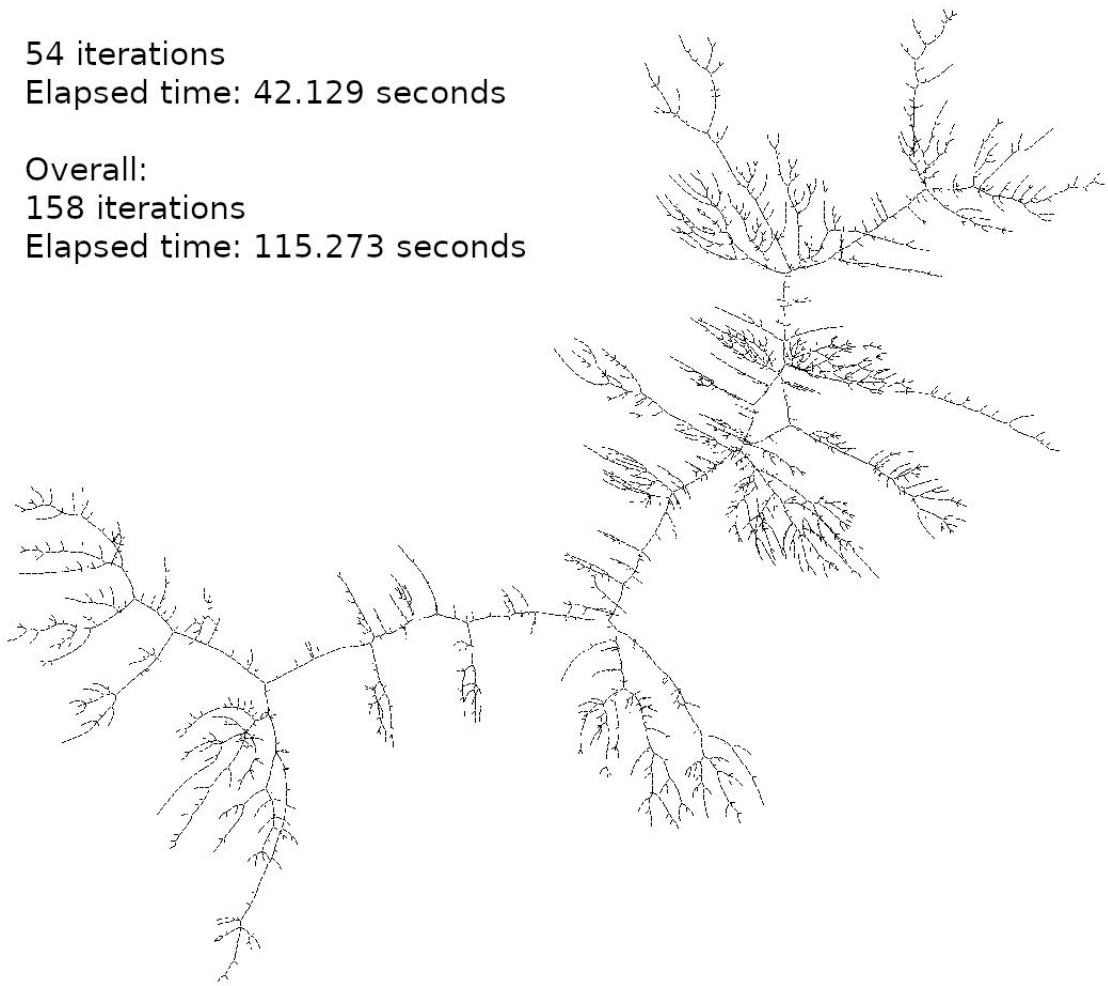
54 iterations

Elapsed time: 42.129 seconds

Overall:

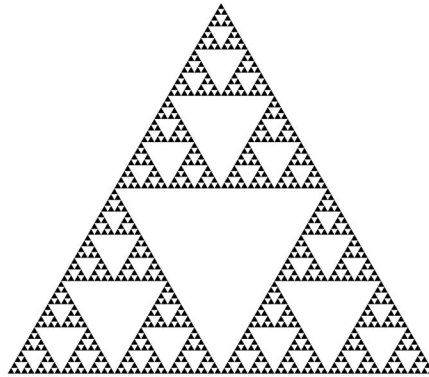
158 iterations

Elapsed time: 115.273 seconds



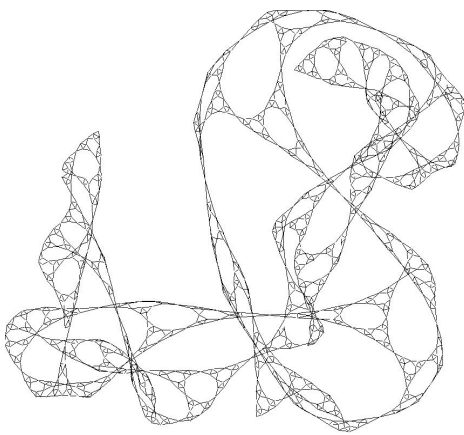
## Comparison to 'Force Directed'

"Force Directed" is a layout algorithm comparable to SM, which already exists in *VANTED*. In the following, SM and ForceDirected are compared with each other. For this we use the example graph 'Sierpinski 6'.

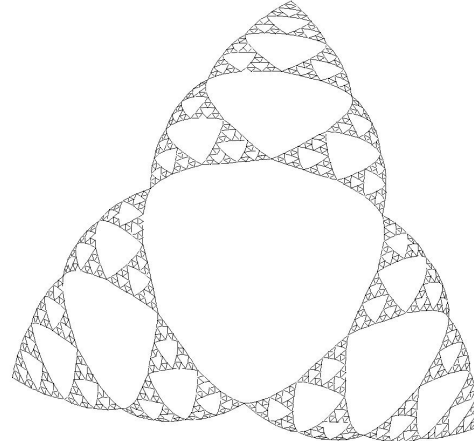


Perfectly layouted 'Sierpinski 6'

Both algorithms were executed in default settings.



'Force Directed'



'Stress Minimization'