User Manual:
Multilevel Framework Add-on for VANTED

—A VANTED Add-on for Layouting Large Graphs using a Multilevel Strategy—

Table of Contents

Introduction

Overview

This add-on adds a multilevel framework to VANTED. It can potentially improve the quality of layouts generated by energy-based layout algorithms (such as the “Force Directed” algorithm that comes with VANTED). This is achieved by applying the algorithm to coarsened (simplified) versions of the graph (known as “levels”), starting with the “coarsest” (most simple) one. The generated layout is then used to compute an initial layout for the next, more detailed, level.

Libraries

This add-on doesn’t require any additional libraries. It does, however, use some of the libraries that are already required by VANTED, such as Apache Commons library.

Copyright

This software is licensed under the same license as VANTED,[1] the “GNU General Public License”. Version 2 of the license is used.[2]

Acronyms and Abbreviations

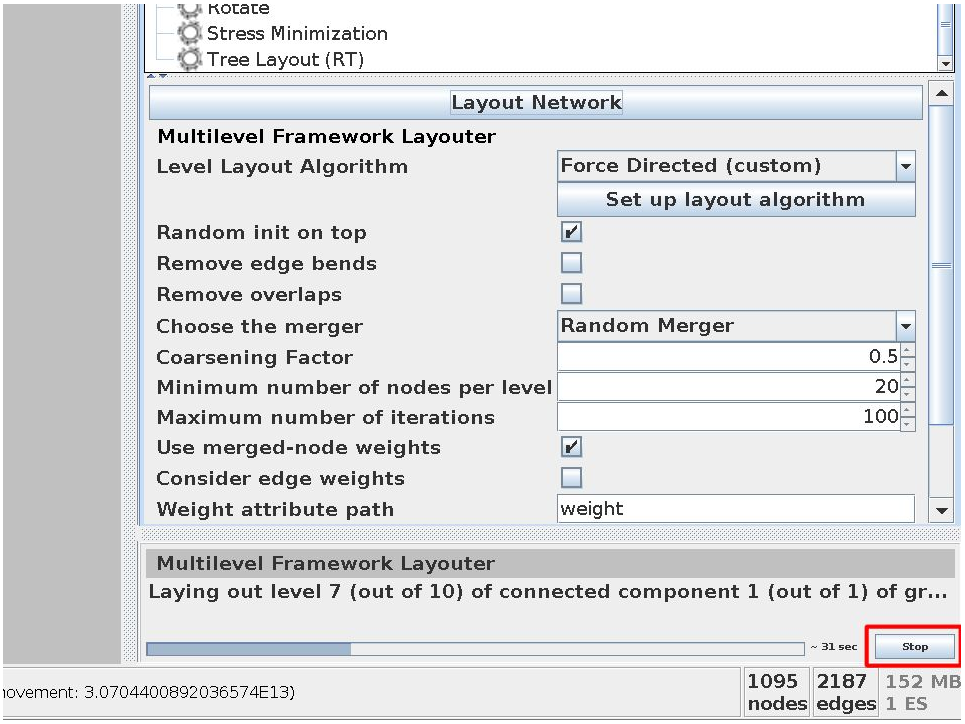
VANTED — Visualisation and Analysis of Networks conTaining Experimental Data
WTFPL — Do What The Fuck You Want To Public License
MLF — Multilevel Framework *(the name of the add-on)*

[1] https://bitbucket.org/vanted_dev/vanted/src/master/src/main/resources/license.txt
[2] https://www.gnu.org/licenses/old-licenses/gpl-2.0.html

Preconditions and Behaviour

The MLF is meant to be used on undirected graphs. Directed graphs will be treated as if they are undirected. It refuses to lay out empty graphs. You cannot start multiple MLF processes that work on the same graph at the same time. You can, however, layout multiple graphs at the same time, *if* the algorithm supports running on multiple graphs using *the same algorithm instance*. To our knowledge, the *only* algorithm available for VANTED that we can guarantee will work in this way is the "Stress Minimization" algorithm developed by Team 2.2 during the Softwareprojekt 2019. [3]
Only selected nodes will be laid out. If no nodes are selected, the layout will be performed on the whole graph. If there is a selection, the other nodes and all edges that connect them to other nodes will be completely ignored.[4]

The layout can be stopped using the "Stop" button that is displayed next to the MLF task (see image below).



Note that the MLF will only stop after the selected algorithm is done laying out the *current* level.

[3] We did not check all algorithms' code. But if the execute() method uses any instance variables at all during the main loop it is probably not going to support executing on multiple graphs at the same time.
[4] (i.e. the MLF works on the subgraph induced by the selected nodes)

Work with the Multilevel Framework

Provided Functions

The add-on provides the option to choose the Multilevel Framework as layouter. The user can choose a "**merger**", choose a "**placer**" and customize their behaviour via the GUI.

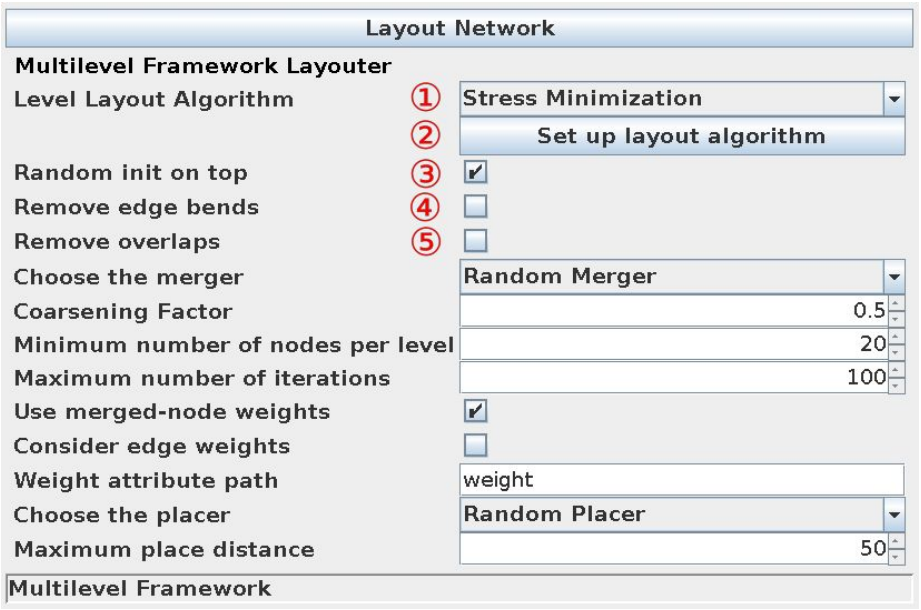
The merger builds multiple graphs ("**levels**") starting from the input graph. This process of creating representative graphs of gradually decreasing size is known as *coarsening* or *merging*. During the merging process, several nodes are merged into one node. This node is referred to as a "**merged node**". The nodes that it represents are referred to as its "**inner nodes**". (Note that the inner nodes can also be merged nodes of lower coarsening levels.)

Starting with the smallest graph ("coarsest level") the chosen layout algorithm is applied. Then the placer uses the layout of the next (less "coarsened") level to form an initial layout for the current layouting step. Traversing downwards through all the levels (from the coarsest to the least coarse level, i.e. the original graph) results in adjusted initial layouts for all layouting steps.

Limitations

Every layouting step starts with an initial layout which should be close to the result and "non-base-levels" are significantly smaller and therefore less expensive to compute. On the other hand for every level an additional graph has to be layouted. As a result depending on the graph, the merger, the placer and their parameters applying the multilevel can have different effects on the performance of the layouter and the quality of the final layout.
This implementation aims to layout graphs with around 4,000 nodes and should be able to handle graphs with up to 10,000 nodes.

General Options



① Here you can choose which algorithm should be applied on each level. We recommend using the “Stress Minimization” algorithm that was already mentioned above, because it provides special support for the MLF.
② Here you can change the parameters of the layout algorithm that will be run on each level.
③ This option will initialize the top level with a random layout before the algorithm is run. If you already have a good starting layout the running time and end layout might improve if you disable this option. It is enabled by default.
④ Edge bends are a feature of VANTED allowing you to create bent edges. Consult the VANTED documentation for information about what they are. The MLF ignores edge bends, which can lead to strange-looking results if you have them in your graph. This is why we provide this option, that simply removes all edge bends before the layout is performed.
⑤ If this option is enabled overlapping nodes will be moved further apart after of the layout of each. If you notice overlapping nodes in your layout, it might be an indication that the layout algorithm you’re using isn’t very good and you should consider changing its parameters or swapping it out for another algorithm.

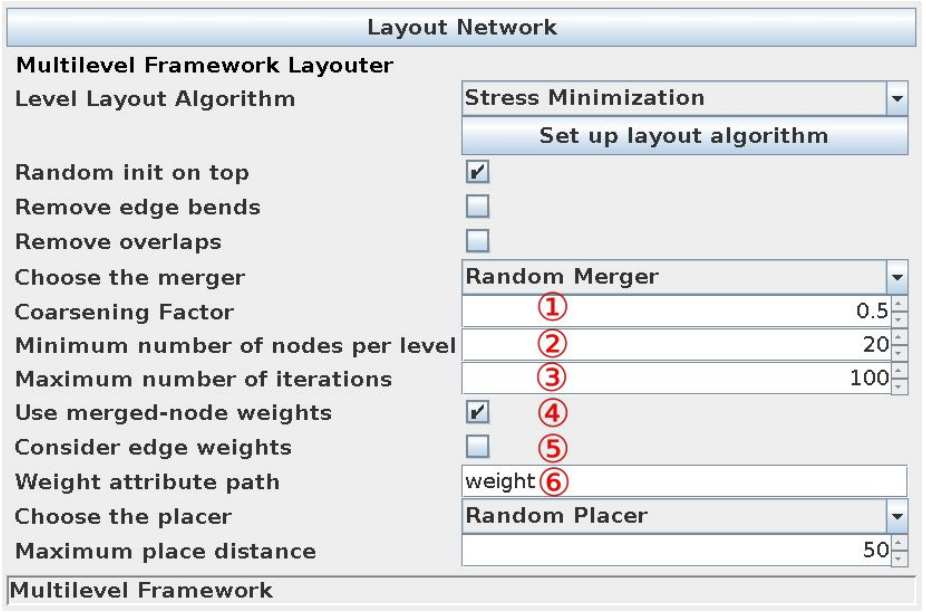
Note that the random layout, the edge bend removal and the node overlap removal is done using the respective algorithms provided by VANTED. Consult the VANTED documentation for more information about what these algorithms do.

Mergers

Random Merger

The random merger selects edges at random and merges the nodes that they connect (the edges are “collapsed”). By default it is not completely random, but prefers merging nodes with low weight (low number of nodes that they contain). It does this until a stop criterion (specified by the parameters described below) becomes true or no edges are left. The random merger stops at the first stop criterion that is hit.
As you will see below, the random merger is not completely “random”. It can for example prefer to merge nodes that don’t already contain lots of inner nodes.

Options



① Determines the fraction of nodes that will be merged into other nodes. (If n is the current number of nodes, and c is the coarsening factor, the next level will contain roughly $n - c \cdot n$ nodes.) The higher the coarsening factor is, the lower the number of coarsening levels will be. Note that his value has to be in the range $(0, 1)$.
② The minimum number of nodes per level is a stop criterion for the random merger. If it finds that the number of nodes of the coarsening level that is currently being generated is less than or equal to this number, it will stop the coarsening process.
③ This number limits how often the coarsening process is repeated (iterated). It limits the number of levels that will be generated. Usually the random merger will stop because of the stop criterion described in ②, but if you want it to stop earlier, you can change this parameter (although increasing the coarsening factor or using a different merger is probably better in this case).
④ This option actually makes the random merger less “random”. If you enable it the random merger will prefer merging edges that connected merged nodes that don’t already have lots of inner nodes starting from the second coarsening level. (The first level

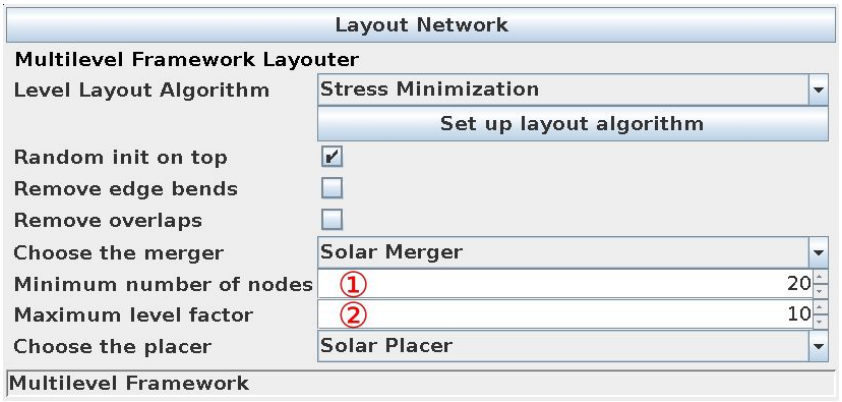
is the original graph and does not consist of merged nodes.) It is enabled by default, as it generally leads to much better layouts.
⑤ If you also want the merging in the first level to work in a similar way as the other levels (see ④), you can enable this option. If this option is enabled, the random merger will prefer merging nodes with low edge weight on the first level.[5] Note that this requires manually or programmatically setting the edge weight. It is probably only useful if the original data already contains pre-generated edge weights.
⑥ This option determines the name of the edge weight attribute that will be used if option ④ is enabled.

Solar Merger

The solar merger treats the graph and all the resulting smaller graphs as galaxies. The latter are a set of solar systems. Every node of the graph takes the role of a stellar body. There are three different types of bodies. Suns being the centers of their respective solar systems, planets which are nodes adjacent to their suns and moons which are neighbors of a planet but do not share edges to any sun.
In each step coarsening step every solar system is “collapsed” into its sun. In this step all edges between nodes of the collapsed systems are kept as edges between the resulting nodes. To fully take advantage of the Solar Merger, you should choose a placer which uses the solar system structure to place the nodes (the “Solar Placer”).
In order to build solar systems for a graph (level) first the graph has to be partitioned. This is done by determining a set of suns. Using a candidate set containing all nodes a random node is marked as a sun. The sun and the nodes which are one and two edges away are removed from the candidate set. This is done until there are no candidates left.
The set of suns is then used to set up solar systems. For each of the suns the neighbors are marked as planets with their sun. The remaining nodes which are neither sun nor planet are marked as moons with a single planet next to them as their planet. As a result all nodes of the Graph have a role (sun, planet or moon) and a corresponding solar system.
Finally nodes and edges for the next level have to be created. For every solar system of the galaxy the sun, all planets and moons are added as inner nodes to a merged node in the new graph. For the new nodes the edges are calculated. If one of the inner nodes had an edge to an inner node of another solar system an edge is created between the two new nodes.
This is process is done until one of the stopping criteria is fulfilled.

[5] The weights can be `int`, `long`, `float` or `double` attributes.

Options



① The minimum number of nodes is a stopping criterion for the Solar Merger. When the number of nodes of the graph/subgraph is lower than the given value the Solar Merger stops.

② The maximum level factor is a stopping criterion for the Solar Merger. It limits how many times a new galaxy is created or in other words how many levels are generated. The size of the original graph divided by the maximum level factor is used as the maximum amount of levels.

Placers

Random Placer

Places nodes randomly in a given radius around the merged node.

Options

The radius within which the Random Placer places the nodes can be configured. If the layout algorithm you chose has a target edge length or a minimum edge length, a reasonable value for the radius option would be half of this length.

Solar Placer

The Solar Placer is the correspondent placer to the Solar Merger. It uses to the structure of solar systems to create promising initial layouts for the layouters. The placement in done by placing the sun for every solar system at position of node representing the solar system in the smaller graph. Planets which do not have neighbors in other solar systems and no moons with neighbors in other solar systems are placed randomly around the sun. Moons which do not have neighbors in other solar systems are placed randomly around their planets. Planets and moons which have neighbors in other solar systems are placed on a path between these solar systems' suns. The planets and moons on such a path are placed equidistantly on a line between these suns.

Options

There are currently no option to customize the Solar Placer's behavior in the GUI.

Recommended Settings

The best configuration from our experience is *"Stress Minimization" + "Solar Merger" + "Solar Placer"*
If you are using the Force Directed algorithm, *"Solar Merger" + "Solar Placer"* will give you better performance, but *"Random Merger" + "Random Placer"* will generally lead to better results.