Guide - Changes

Volker Maag, Thomas Willhalm

9. August 2002

1 Guide - short description

Guide is a program to visualize large graphs[1] efficiently. It has an interactive display to visualize the graph and it is based on the previous version named GRIP. It can read files in the gml-format and there are some test graphs like grid and moebius band implemented. The algorithm uses an intelligent clustering of the vertices in several layers and a force-based model to arrange the vertices in a layer in two, three or four dimensions. The force-based model is based on the method of Kamada-Kawai. Only in the last layer the method of Fruchtermann-Reingold is used.

You may find the original source code, documentation and user's guide on `http://www.cs.arizona.edu/people/kobourov/GRIP/`

2 Extensions

The following extensions were implemented:

1. read and write support for the GraphML-format[2]
2. support for weighted graphs

3 Algorithmic Details

3.1 Improvements

While testing it turned out that two-dimensional grids with more than 100 vertices were not drawn as smooth as possible but had some waves when embedded in three dimensions. This undesirable behaviour was explored with the intention to eliminate it. However, the algorithm could not be improved significantly.

3.2 Support for weighted graphs

In the original version edges had only constant weight. By manipulating this constant the drawing could be zoomed. Now the data structure for the graph is extended to handle weighted edges. Furthermore, some weighted test-graphs were implemented. It was neccessary to adjust the algorithm in following points:

- Replacement of the simple breadth first search by Djikstra's algorithm

---

[1] On an AMD Athlon 1400MHz/256MB RAM it worked with a graph with about 25 000 vertices.

[2] For an up-do-date documentation of the GraphML-standard look at `http://www.graphdrawing.org/graphml/index.html`

- Adjustment of some parameters in the calculation of the Fruchtermann-Reingold-/ Kamada-Kawai-vector. The new formulas look like:
  Kamada-Kawai[3]:

$$disp(u) = \sum_{\substack{\text{all vertices } v\\ \text{in the current layer}}} \left(\frac{\|v-u\|^2}{dist(v,u)^2} - 1 \right) (v-u)$$

  Original formula:

$$disp(u) = \sum_{\substack{\text{all vertices } v\\ \text{in the current layer}}} \left(\frac{\|v-u\|}{dist(v,u)w} - 1 \right) (v-u)$$

  Fruchtermann-Reingold:

$$disp(u) = \underbrace{\sum_{\substack{\text{all vertices } v\\ \text{which are adjazent to } u}} \left(\frac{\|v-u\|^2}{dist(v,u)^2} - 1 \right) (v-u)}_{\text{attractiv force}} - \underbrace{\sum_{\substack{\text{all vertices } v\\ \text{in the current layer}}} \left(\frac{0.05l^2}{\|u-v\|^2} \right) (v-u)}_{\text{repulsiv force}}$$

  Original formula:

$$disp(u) = \sum_{\substack{\text{all vertices } v\\ \text{which are adjazent to } u}} \left(\frac{\|v-u\|^2}{w^2} - 1 \right) (v-u) - \sum_{\substack{\text{all vertices}v\\ \text{in the current layer}}} \left(\frac{0.05w^2}{\|u-v\|^2} \right) (v-u)$$

  where u, v are the position vectors of the corresponding vertices, $disp(u)$ states the direction in which the vertex u is moved during an iteration[4], $\|v-u\|$ is the current distance in the coordination system, $dist(v,u)$ the required distance given by the weight of the edges, l is the average edge weight and $w = 32$ the constant edge weight of the original version. The scaling factor 0.05 was taken from the original version; this value seems to provide the best results. In the calculation of $disp(u)$ according to Kamada-Kawai, $dist(v,u)$ is simply replaced by the given edge length instead of the constant. However, the calculation of the attractive force of Fruchtermann-Reingold now considers $dist(v,u)$ (which it did not before).

No changes were made in the calculation of the layers or of the local temperature.

4 Technical Details

4.1 Source Code Overview

File	Description
main.c	main program - combines creation/reading of the graph and drawing
gml_*.c	implementation of gml-file-reader (converter, demo, parser, scanner)

---

[3] The formula is also used when a vertex is placed for the first time but then only the nearest three vertices in the current layer are considered instead of all vertices.

[4] The vector $disp(u)$ is normalized before use; only the change of the direction compared to the previous iteration influences the amplitude strength.

GraphMLConverter.cc	implementation of GraphML-file-reader written in C++ because of the xerces library
GraphMLException.cc, handlers.cc	handles parsing errors for GraphML-files
GraphMLNames.cc, GDNames.cc, GD.h	definition of the standard GraphML-tags and their extensions for handling the weight of an edge and the position of a vertex
GraphMLDocument.cc, GraphML.h, DOMAdmin.cc/.h	further tools for the GraphML-Converter
cplusplusinterface.h	declaration of functions written in C++ that are called from c
bfs.c/.h, dijkstra.c/.h	implementation of a graph breadth-first-search- and Dijkstra's algorithm
Queue.c, PQueue.c, Heap.c	implementation of a queue, a prioritiy queue and a heap (which is used by the priority queue) needed for bfs and Dijkstra
Point.c/.h	implementation of a point-data structure
Graph.c	implementation of test graphs like trees, (un)weighted grids, cubes, ...
Graph.h	definition of the main data structure "graph"
misf.c/.h	creation of the misf-layers
DrawGraph.c	implementation of the vertex-positioning algorithms
MesaPlot.c/.h	implementation of plotting and user interface specific tasks
graph.tcl	tcl-user interface – makes it easy to set the parameters for the program
help.data, help.tcl	online help for the tcl-interface
graphml-structure-1.0rc.xsd	schema definition of the GraphML-Format (it is not the original version but a modified version that works)[5]
Makefile	for building the source code
readme	some information for building and executing the program

4.2 Requirements

- the programm was tested on Linux SuSE 8.0/7.3-Systems but should work with other distributions as well
- xerces version 1.7.0 (1.6.0 might work too but you need to adjust include paths in the source code)
- an OpenGL-library (e.g. MesaGlu)

4.3 Building

You have to set the correct paths to your installation of xerces in the Makefile. After that type `"make"`.

4.4 Running

Probably you have to adjust the environment variable `LD_LIBRARY_PATH` to include the xerces library.[6]

Then type `"graph.tcl"` or `"wish graph.tcl"` to set the parameters with an interface or start `main` directly setting the parameters "by hand". If results are written, you can find them in the file `erg.xml`.

---

[5] The changes in the schema definition do not affect the format of the XML-file.

[6] e.g. `"export LD_LIBRARY_PATH="/home/student/maagv/hiwi/xerces-c1_6_0-linux/lib:$LD_LIBRARY_PATH"`

You may want to use the help provided by graph.tcl

4.5 Parsing GraphML-Files

The validation of the xerces parser works only with an modified version of the GraphML schema, but you can switch off validation by compiling with the compiler option -DNO_XERCES_VAL. The program reads the schema style GraphML format but it should be able to validate the former dtd style format, too. While writing, the program tries to keep up all information it read (if it read a GraphML-file) but e.g. the direction property will be lost. Furthermore it may be suitable for large graphs to work with GraphML files with no further information because all this information will be kept in memory, so less memory will be available for computing. Lack of memory is not caught and will propably result in a segmentation fault.

4.6 Specification of the allowed graphs to be parsed

The programm accepts GraphML-graphs with the following properties:

- undirected graphs
- directed graphs; RESTRICTION: will be converted in undirected graphs (for computing and writing)
- weighted, unweighted or partially weighted edges with a given standard weight (If the standard weight is missing, the value 32 is used.)
- graphs with loops and double edges; RESTRICTION: these will be lost (for computing and writing)
- graphs with additional information; REMARK: the GraphML-format allows for every element (edge, node, the graph itself) of the graph to have additional information. This information should be kept when saving the graph with its new coordinates under the restrictions noted above (but this feature was never really completely tested).

Further restrictions:

- Only the first graph in a file will be handled.

The GML-Graphs might have more properties with less restrictions, see the documentation for original Guide.

4.7 Specification of the extended GraphML-format

The standard GraphML-format is not changed but extended in order to allow the weight of edges and the coordinates of vertices to be saved.

If you want to set a weight of the edges, insert the following lines after the GraphML tag:

```
<key id="we" for="edge" attr.name="weight" attr.type="double">
<default>default weight of edge</default>
</key>
```

The number enclosed by the tag `default` defines a default weight. Setting a default weight is optional. To set the weight of an edge, insert a data-tag to an edge-tag like this:

```
<edge source="v0" target="v1">
<data key="we">weight of edge</data>
</edge>
```

The weight of an edge can be given as an integer or a real value.

The coordinates are stored like this:

```
<node id="v0">
<data key="x1"> 50.1536 </data>
<data key="x2"> 168.311 </data>
<data key="x3"> -62.3714 </data>
</node>
```

If there are four or two coordinates, four or two coordinates will be saved. If there are coordinates in the file, the following key-tags must also be present:

```
<key id="x1" for="node" attr.name="x1-coord" attr.type="double"/>
<key id="x2" for="node" attr.name="x2-coord" attr.type="double"/>
<key id="x3" for="node" attr.name="x3-coord" attr.type="double"/>
```