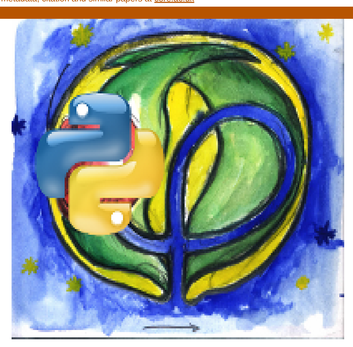


Graph analysis and visualization with Tulip-Python

Antoine Lambert, David Auber
antoine.lambert@labri.fr, david.auber@labri.fr

The Tulip framework

Graphs play an important role in many research areas, such as biology, microelectronics, social sciences, data mining, and computer science. Tulip [1] is an information visualization framework dedicated to the analysis and visualization of such relational data. Written in C++ the framework enables the development of algorithms, visual encodings, interaction techniques, data models, and domain-specific visualizations.

Tulip-Python bindings

The Tulip framework is now available to the Python community through the Tulip-Python bindings. The bindings have been developed using the SIP tool [4], allowing to easily create quality Python bindings for any C/C++ library. The main features provided by the bindings are the following ones:

- **Creation and manipulation of graphs :** Tulip provides an efficient graph data structure for storing large and complex networks. It is also one of the few that offer the possibility to efficiently define and navigate graph hierarchies or cluster trees (nested sub-graphs).
- **Storage of data on graph elements :** Tulip allows to associate different kind of serializable data (boolean, integer, float, string, ...) and visual attributes (layout, color, size, ...) to graph elements. All these data can be easily accessed from the Tulip graph data structure facilitating the development of algorithms.
- **Application of algorithms of different types on graph :** Tulip has been designed to be extensible and provides a variety of algorithms (layout, clustering, ...) implemented as plugins. These algorithms can now be easily called from Python. As Tulip is dedicated to graph visualization, it is provided with numerous state of the art graph layout algorithms (especially a bridge to the Open Graph Drawing Framework [2]).
- **Creation of interactive visualizations :** Tulip OpenGL visualizations (typically node-link diagrams) can be created from Python. Visualizations are synchronized, meaning every modification on the visualized data (graph structure, visual attributes, ...) triggers automatic redraw.
- **The ability to write Tulip plugins in pure Python :** Python developers can now contribute to Tulip and write plugins (algorithms, graph import/export) in their favorite language. These plugins can be called and integrated in the Tulip software like the C++ ones.

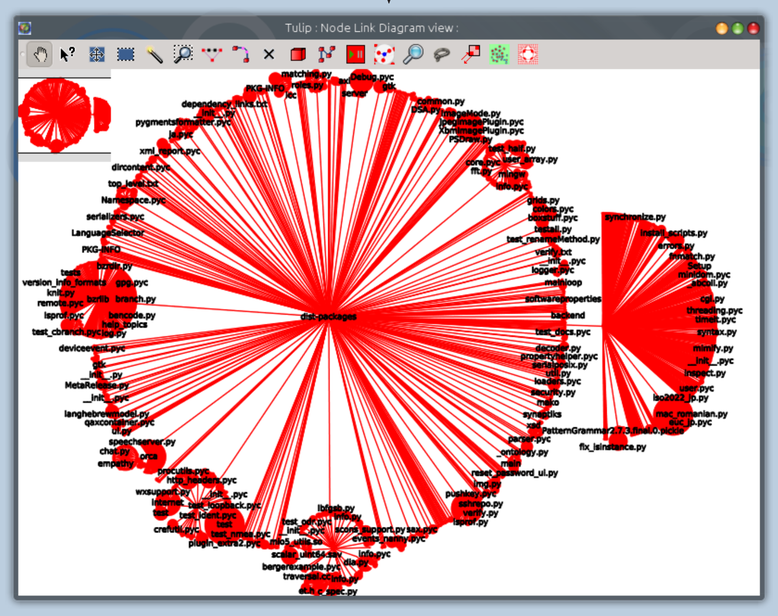
Interactive use

Tulip bindings can be used through the classical Python shell (script and interactive mode) and give access to all graph algorithms and visualizations available in Tulip.

Importing, customizing and visualizing a network can be done in a few lines of code, as illustrated by the snippet below which creates a visualization of the Python 2.7 library folder.

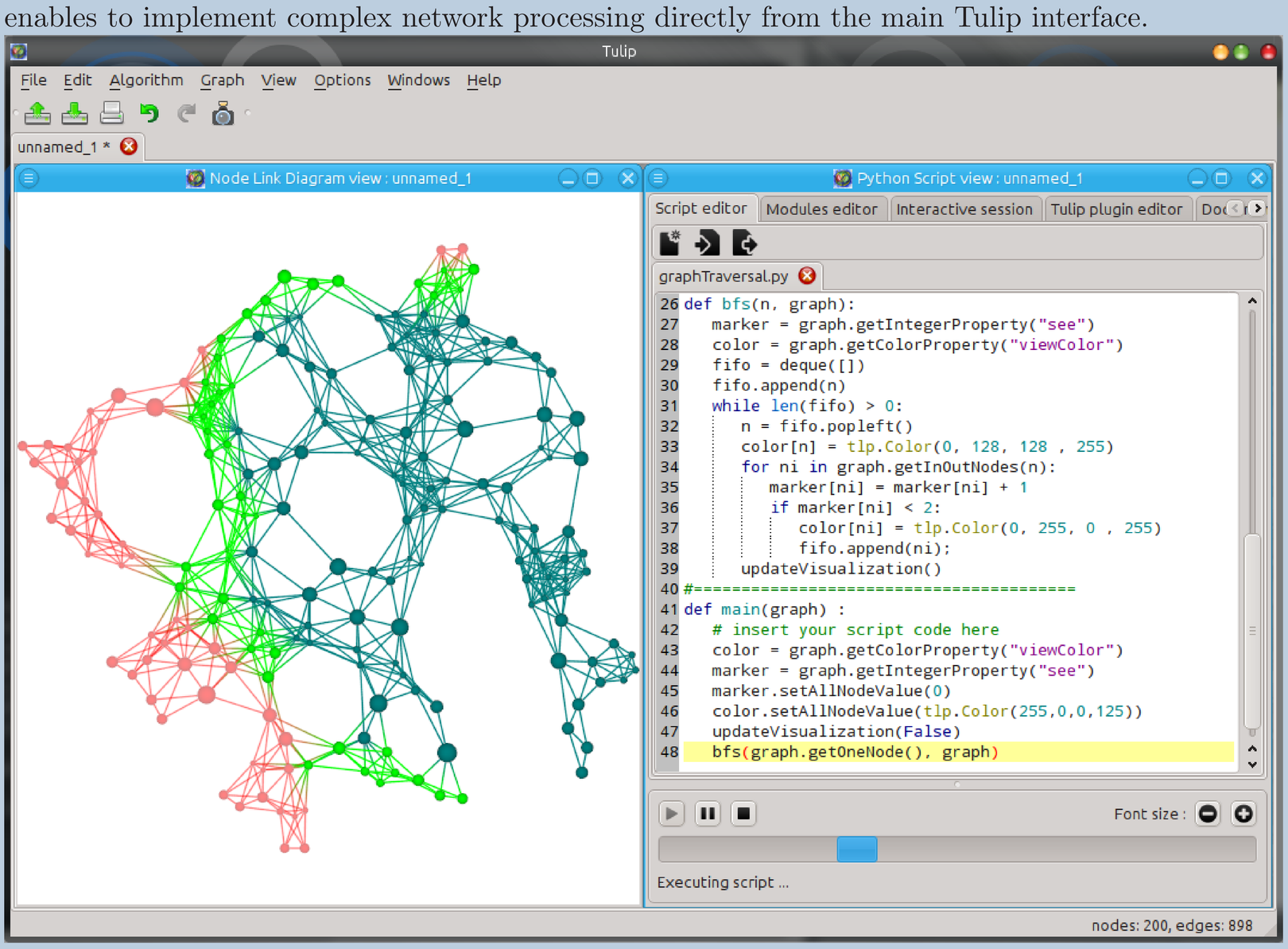
```
>>> from tulip import *
>>> from tulipgui import *
# Call the File System import plugin on the Python 2.7 library folder
>>> params = tlp.DataSet()
>>> params["dir::directory"] = "/usr/lib/python2.7"
>>> graph = tlp.importGraph("File System Directory", params)
# Set nodes labels to filenames
>>> graph["viewLabel"].copy(graph["name"])
# Draw the graph
>>> graph.applyLayoutAlgorithm("Bubble Tree", graph["viewLayout"])
# Visualize the graph
>>> view = tlp.addNodeLinkDiagramView(graph)
>>> rp = view.getRenderingParameters()
>>> rp.setLabelsDensity(0)
>>> view.setRenderingParameters(rp)
```

⇓



Using the bindings from the Tulip software GUI

A lightweight Python IDE is now integrated to the Tulip software giving the ability to write Python scripts and execute them on graphs visualized in Tulip. The benefits of integrating Python to Tulip were numerous and various : interactive modification of the graph structure/visual attributes, custom graph and data import/export, prototyping and chaining of algorithms, bridging Tulip and other Python Graph modules like NetworkX [3], ... That scripting feature also turns Tulip into a powerfull tool for teaching graph theory, graph mining and graph visualization. The power of Python enables to implement complex network processing directly from the main Tulip interface.



References

[1] D. Auber, D. Archambault, R. Bourqui, A. Lambert, M. Mathiaut, P. Mary, M. Delest, J. Dubois, and G. Mélançon. The Tulip 3 Framework: A Scalable Software Library for Information Visualization Applications Based on Relational Data. Technical report RR-7860, INRIA, January 2012. `http://www.tulip-software.org`.

[2] M. Chimani, C. Gutwenger, M. Jünger, K. Klein, P. Mutzel, and M. Schulz. The Open Graph Drawing Framework. 15th International Symposium on Graph Drawing 2007, Sydney (GD07), 2007. `http://www.ogdf.net`.

[3] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[4] Riverbank Computing Limited. SIP - a tool for automatically generating Python bindings for C and C++ libraries. `http://www.riverbankcomputing.co.uk/software/sip/`.