Graph analysis and visualization with Tulip-Python

Antoine Lambert, David Auber

EuroSciPy 2012 - 5th European meeting on Python in Science
Brussels, Université libre de Bruxelles, August 23-27 2012

Graphs analysis and visualization with Tulip-Python

A. Lambert[1] and D. Auber[1]

[1]CNRS UMR5800 LaBRI, Université Bordeaux 1
and INRIA Bordeaux - Sud Ouest, France

Graphs play an important role in many research areas, such as biology, microelectronics, social sciences, data mining, and computer science. Tulip (`http://www.tulip-software.org`) [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.

The Tulip core library is now available to the Python community through the Tulip-Python bindings. The bindings have been developed using the SIP tool [4] from Riverbank Computed Limited, 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 easily extensible and provides a variety of graph algorithms (layout, metric, clustering, ...) implemented as C++ plugins. All 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 but also a bridge to the Open Graph Drawing Framework [2].
- **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.

The bindings can be used through the classical Python shell but were primarily designed to add a scripting feature to the Tulip software. Since the 3.5 release, a lightweight Python IDE has been added to the Tulip interface giving the ability to write Python scripts and execute them on graphs visualized in Tulip (see Figure 1). The benefits of integrating Python to Tulip are numerous and various : interactive modification of the graph structure/visual attributes (triggering update of visualizations), 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.

Only the Tulip core library (data model) has been wrapped for the moment but we are currently working on the development of bindings for the Tulip OpenGL and the Tulip Qt library. These new modules will enable to customize the Tulip OpenGL scenes but also to develop new visualization components and interaction techniques for Tulip in pure Python.

References

[1] David Auber, Daniel Archambault, Romain Bourqui, Antoine Lambert, Morgan Mathiaut, Patrick Mary, Maylis Delest, Jonathan Dubois, and Guy Mélançon. The Tulip 3 Framework: A Scalable Software Library for Information Visualization Applications Based on Relational Data. Technical report RR-7860, INRIA, January 2012.

[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.

[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/`.

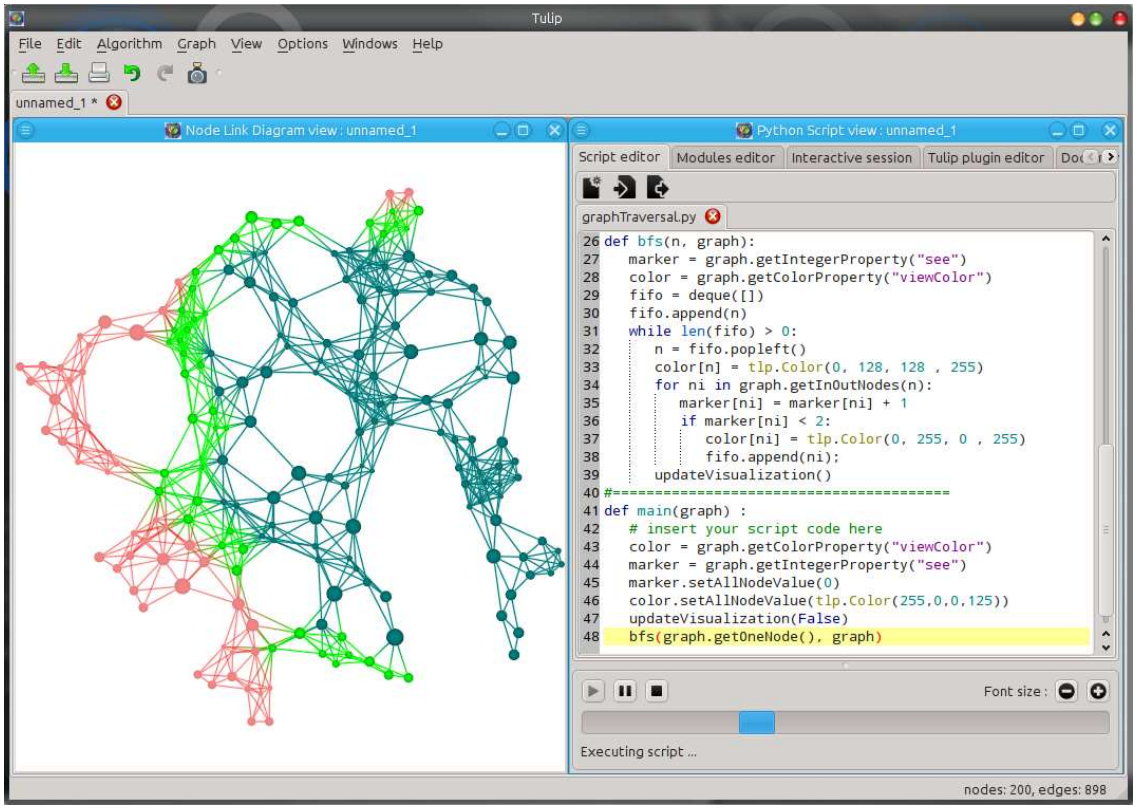

Figure 1: A Python script executing a BFS traversal on the graph visualized in Tulip. The script modifies the color of the nodes the following way : red for unvisited nodes, blue for treated nodes, green for currently visited nodes.