ProViz: protein interaction visualization and exploration

Florian Iragne, Macha Nikolski, Bertrand Mathieu, David Auber and David Sherman∗

LaBRI UMR 5800, Université Bordeaux 1, 351 cours de la Libération, 33405 Talence Cedex, France

Received on August 4, 2003; revised on June 15, 2004; accepted on August 18, 2004
Advance Access publication September 3, 2004

ABSTRACT
Summary: ProViz is a tool for the visualization of protein–protein interaction networks, developed by the IntAct European project. It provides facilities for navigating in large graphs and exploring biologically relevant features, and adopts emerging standards such as GO and PSI-MI.
Availability: ProViz is available under the GPL and may be freely downloaded. Source code and binaries are available at http://cbi.labri.fr/eng/proviz.htm
Contact: david.sherman@labri.fr

INTRODUCTION
Analysis of protein–protein interaction (PPI) networks requires a combination of algorithmic and visualization tools, ideally integrated within a software platform that is itself integrated with access to local and distant data banks. We present a software tool called ProViz that provides highly interactive visualization of large networks of interactions, integrated with the IntAct data model (Hermjakob et al., 2004a). ProViz is similar in purpose to PIMrider (Legrain et al., 2001), Osprey (Breitkreutz et al., 2003), and other visualization or analysis tools (Tucker et al., 2001; Lappe et al., 2001; Koike and Rzhetsky, 2000; Shannon et al., 2003).

OVERVIEW OF ProViz
Graph drawing and interactive graph exploration are active domains in computer science and many tools are available for this task. Adaptation of these tools and techniques to the specific needs of biologists exploring PPI networks is a current effort in bioinformatics. The challenge is to add valuable information and functions that enable the user to discover interesting biological relations hidden within the data.

ProViz improves over existing work by providing a fast, scalable, open tool with extensive plugins, that integrates emerging standards for representing biological knowledge in a biologist-oriented interface.

∗To whom correspondence should be addressed.
Fig. 1. View of a spoke-model PPI graph for yeast. (a) Protein properties. (b) Protein selection using GO terms. (c) Main window showing part of current view (right) and the cluster tree (left).

Layout algorithms. Of the dozens of layout algorithms in the plugin library, three were chosen for direct use based on their capacity to highlight biologically pertinent information. GEM (Frick et al., 1994) is an efficient directed force-based graph drawing algorithm. It groups related nodes and can be used to quickly identify proteins with a given role, or for visualizing protein complexes. Hierarchical layout (Messinger et al., 1991) reveals ancestral relationships between nodes and is useful when looking for cascade-type interactions or comparison to metabolic pathway data. Circular layout is a neutral choice that does not attribute any semantics to edge relations.

Integrating controlled vocabularies. ProViz uses GO and PSI-MI controlled vocabularies for describing proteins and interactions. Users employ these vocabularies when building views of interaction networks by manual filtering or through the use of clustering plug-ins. In Figure 1, we see the property list for the node corresponding to yeast Rad16 (nucleotide excision repair protein), including GO evidence, gene names and external links.

Tulip development platform. ProViz development is based on the Tulip platform (Auber, 2003), designed for management and three-dimensional display of large graphs. It provides a rich set of operations on graphs: metric computation, node and edge layout, selection, extraction of view and subgraphs, and labeling of nodes and edges with arbitrary sets of attributes. Operations specific to the application domain are provided by means of software plugins. Any program using Tulip can add to the core features by providing its own domain-specific plugins. Tulip is written in C++ and uses Qt and OpenGL for enhanced portability.

ACKNOWLEDGEMENTS

This work is supported by EU grant number QLRI-CT-2001-00015 under the RDD programme ‘Quality of Life and Management of Living Resources’.

REFERENCES


