BIOINFORMATION
Discovery at the interface of physical and biological sciences

open access www.bioinformation.net Web Server

Volume 6(7)

OntoVisT: A general purpose Ontological Visualization Tool

Alok Kumar Srivastava*, Narinder Singh Sahni

School of Computational and Integrative Sciences, Jawaharlal Nehru University, New Delhi - 110067, India; Alok Kumar Srivastava - Email: foralok@gmail.com; *Corresponding author

Received May 21, 2011; Accepted June 06, 2011; Published June 23, 2011

Abstract:
Ontologies have emerged as a fast growing research topic in the area of semantic web during last decade. Currently there are 204 ontologies that are available through OBO Foundry and BioPortal. Several excellent tools for navigating the ontological structure are available, however most of them are dedicated to a specific annotation data or integrated with specific analysis applications, and do not offer flexibility in terms of general-purpose usage for ontology exploration. We developed OntoVisT, a web based ontological visualization tool. This application is designed for interactive visualization of any ontological hierarchy for a specific node of interest, up to the chosen level of children and/or ancestor. It takes any ontology file in OBO format as input and generates output as DAG hierarchical graph for the chosen query. To enhance the navigation capabilities of complex networks, we have embedded several features such as search criteria, zoom in/out, center focus, nearest neighbor highlights and mouse hover events. The application has been tested on all 72 data sets available in OBO format through OBO foundry. The results for few of them can be accessed through *OntoVisT-Gallery*.

Availability: The web based application can be accessed through http://ccbb.jnu.ac.in/OntoVisT.html. The applet for visualization is also downloadable from the OntoVisT website.

Keywords: ontology; visualization tool; directed acyclic graphs; web server; gene ontology.

Background:
An ontology is a shared conceptualization of knowledge in a particular domain. Biological ontologies define the basic terms and the relations in biological domains, as well as rules for combining these terms and relations. The OBO Foundry **[1]** and BioPortal **[2]** have developed a large library of bio-medical ontologies. These ontologies can be shared across different bio-medical domains. The OBO file format is the most common file format in the OBO Foundry collection. It aims to achieve human readability, ease of parsing, extensibility and minimal redundancy. Out of the 96 ontologies that are currently available through OBO Foundry, 72 are in OBO format. The number of terms in any ontology may vary from a few to several thousand making it difficult to interpret the data even after parsing the OBO files. Visualization is one of the best alternatives to understand the relations contained in any ontology. However, visualization of an ontology is not an easy task. Relationships between the ontology terms are usually represented as Directed Acyclic Graphs (DAG). It is enriched with role relations among concepts and each concept has various attributes related to it. The structure of any ontology allows a node to have multiple ancestors and/or children.

There are several popular approaches available for ontological navigation. The most simple approach is to present them in tabular form (e.g., FatiGO **[3]**), but has the drawback that it loses substantial information regarding the relations between the terms. The second approach is to present the structure in a tree form. There exists two popular ways to represent tree structure: a flat tree (e.g., TO-GO **[4]**) and an expandable tree in which a node with several parents is represented in the tree multiple times, once under each parent (e.g., AmiGO **[5]**). The third approach is to represent the result in the form of static graphs, either as a tree or DAG (e.g., GO Term Finder **[6]**). The static graph structure usually requires less memory in the case of small networks with fewer edges, but it reduces the interpretiability of the complex networks due to cluttered edges and overlapping nodes. In a typical case, the different ontology terms are distributed across different levels of hierarchy. Identifying the clusters of similar terms in the ontology network becomes much simpler and informative, if the results are presented interactively in a DAG structure. One of the problems in displaying large DAG network is the loss of hierarchical order needed for displaying parent-child relationships. The problem with non-hierarchical representation is that its complexity increases with increase in number of nodes in the graph, which makes the interpretation difficult. Alternatively, we can improve the interpretability by modifying the structure in DAG hierarchical form, in such a way that it contains both the property of DAG as well as placement of all child nodes below their parent nodes. The advantage of this representation is that the flow of information is uni-directional reducing the effort to interpret the network.

Implementation:
Application Description:
OntoVisT Visualization module is written in Java using prefuse-alpha **[7]** library. **Figure 1** shows the schematic diagram and work flow of OntoVisT applications. The visualization module can be used in any of the three forms, (1) web server application (2) stand-alone application and (3) plug-in application.

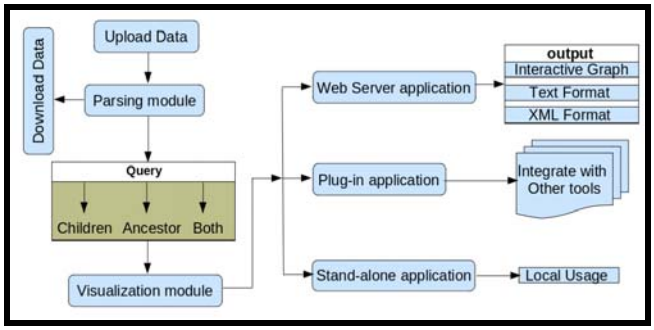

Figure 1: Schematic diagram and work flow of the OntoVisT application. The graph displays the schematic diagram and work flow of OntoVisT.

Web server application:
The web application is divided into two main modules: the parsing module and the visualization module. Users can select any of the 72 ontology files available from the drop down list or upload an ontology file in OBO format. The query form on the same page allows the user to define custom level of abstraction at various levels of hierarchy by selecting the query node, as well as number of levels of the children and/or ancestor. The query node must contain both components: an ontology identifier and a unique id for each ontology (e.g., GO: 0009117, DOID: 00000003, AAO: 00000001). The level defines the depth of the children or ancestors in the hierarchy. Three types of visualization modules have been embedded to explore any ontological structure: (1) Visualization of the *children* of a specific node: a recursive breadth-first search in top-down manner (up to defined level) is carried out to determine all children; (2) Visualization of an *ancestor* of a query node: a recursive breadth-first search in down-top manner (up to defined level) is carried to determine all possible paths back to the top nodes; (3) Visualization of *both*, children as well as ancestors of a specific node. The network is obtained by enumerating all the paths contained in children and ancestor components.

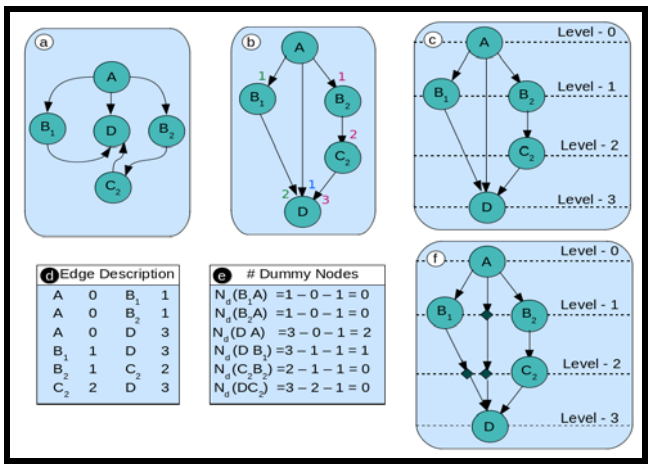

Figure 2: Dummy nodes description. The Figure explains the insertion of dummy nodes for maintaining DAG hierarchical structure. (a) A sample DAG graph with 5 nodes (A, B_1, B_2, C and D). (b) An alternative representation of the same graph indicates three possible routes between A and D. (c) Level assigned to each node is the maximum level assigned when traversed through all the possible paths. (d) Describes the edge description for all paths. The column represents source node, level of source node, sink node and level of sink node respectively. (e) Dummy nodes added to each edge are calculated as the difference between sink and source level minus 1. (f) A unidirectional DAG hierarchical graph after inserting the dummy nodes to the original graph.

OntoVisT Output:
Given the input parameters, OntoVisT provides interactive visualization of any complex ontological network. Complexity increases with the increase in number of nodes in the network. To deal with this problem, we have embedded a search option which highlights the node of interest. The application allows the user to translate the network in such a way that the node of interest is placed in the center by clicking over it. It also allows the user to zoom in/out the topology of the network by using right mouse button. Left mouse button can be used to drag complete network topology. Further, a mouse hover over any node displays the details associated with the term on the bottom left side of the page. The projection of the graph is fixed in order to maintain the consistency in the layout. In order to maintain the hierarchical structure of a DAG, we have inserted dummy nodes in the network, explained with the help of a synthetic example discussed in **Figure 2**. **Figure 2a** shows a small DAG network, and **Figure 2b** shows an alternate and much clearer representation of all possible paths between A and D in the same DAG. Each path assigns a different level (actual level) to each node. We assign a dummy level to each node as the maximum level that occurs when we traverse through all possible paths starting from the root node **(Figure 2c)**. The difference between the dummy levels of the source and sink nodes of each edge, is used to count the number of dummy nodes (N_d) required to maintain the unidirectional flow **(Figure 2d)**. N_d equals difference between sink and source level, minus 1 for each edge **(Figure 2e)**. This leads to a unidirectional DAG hierarchical structure obtained after inserting the desired dummy nodes to each path **(Figure 2f)**.

The standalone application:
The OntoVisT module is platform independent and can be run as a standalone application on any machine with java (version 1.5 or higher) enabled web browser. The source code and the class files for the application are packaged into a single executable jar file OntoVisT.jar. The OntoVisT package, sample XML file, and main html file (OntoVisT.html) can be downloaded from the *OntoVisT download* page. Pre-requisit, installation instruction and usage can be found on the *OntovisT help* page. The visualization module can be used to explore the ontology graph locally through web-browser using OntoVisT.html.

Plug-in application:
OntoVisT applet can be easily integrated as a plug-in with any other ontology based navigation application, requiring only the XML file as an input. The visualization output can easily be modified by modifying the XML file only.

Discussion:
Table 1 (see Supplementary material) presents a comparative summary of OntoVisT with other Java based applications (TGVizTab **[8]**, OntoTrack **[9]**, BINGO **[10]**, GOLEM **[11]**) used for visualizing ontologies. Amongst the surveyed tools, *OntoVisT* is the only tool that can be used to study the modular structure of any ontological network. The supporting files containing DAG structure are downloadable in both text and XML formats and can be further used for custom level of ontological analysis. The OntoVisT applet supports interactive visualization of both trees as well as DAGs. It offers flexibility in terms of its usage, which only requires XML file as an input in a prescribed format.

Authors' contributions:
AKS has conceptualized the study, wrote the application, implemented the web server, tested the program and drafted the manuscript. NSS helped in enhancing the features and writing the manuscript.

Acknowledgments:
AKS is a recipient of a senior research fellow grant provided by CSIR (grant 9/263-0771-9). The author would like to acknowledge COE funding for the school provided by Department of Biotechnology, India.

References:
[1] Smith B *et al. Nat Biotechnol.* 2007 **25**: 1251 [PMID: 17989687]
[2] Noy NF *et al. Nucleic Acids Res.* 2009 **37**: W170 [PMID: 19483092]
[3] Al-Shahrour F *et al. Bioinformatics* 2004 **20**(4): 578 [PMID: 14990455]
[4] Yu U *et al. Bioinformatics* 2005 **21**(17): 3580 [PMID: 15994194]
[5] Carbon S *et al. Bioinformatics* 2009 **25**(2): 288 [PMID: 19033274]
[6] Boyle EI *et al. Bioinformatics* 2004 **20**(18): 3710 [PMID: 15297299]
[7] http://prefuse.org/
[8] http://users.ecs.soton.ac.uk/ha/TGVizTab/TGVizTab.htm
[9] http://www.informatik.uni-ulm.de/ki/ontotrack/
[10] Maere S *et al. Bioinformatics* 2005 **21**: 3448 [PMID: 15972284]
[11] Sealfon RS *et al. BMC Bioinformatics.* 2006 **7**: 443 [PMID: 17032457]

Edited by P Kangueane
Citation: Srivastava & Sahni. Bioinformation 6(7): 288-290 (2011)

BIOINFORMATION

open access

Supplementary material:

Table 1: Comparative analysis of OntoVisT with other interactive tools which displays DAG layout

Application Description	Tool	**TGVizTab**	**OntoTrack**	**BINGO**	**GOLEM**	**OntoVisT**
	Year	2003	2004	2005	2006	2011
Application Type	web based	No	No	No	Yes	Yes
	Stand-alone	Yes	Yes	Yes	Yes	Yes
	used as plug-in	No	No	No	No	Yes
Prerequisite	Base Language	java	Java	Java	Java	java
	Plug-in Required	Protégé	Racer server	cytoscape	No	No
Interactivity	Hierarchical display	No	Yes	Yes	No	Yes
	Depth Calculation	Yes	Yes	No	No	Yes
	Zoom	Yes	Yes	Yes	No	Yes
	Focus	highlight	highlight	highlight	highlight	highlight
	Overlapping nodes	Yes	No	No	No	No
	Projection	Not predictable	Fixed	Fixed	Not predictable	Fixed
Flexibility	Data Bound	No	No	GO bound	GO bound	No
	Bound with Analysis Tool	No	No	Yes	Yes	No
	Custom level abstraction	No	No	No	No	Yes
	Usage	Multiple	Multiple	Multiple	Fixed	Multiple
Supporting file	Format	XML	XML	GML, XGMML	No	Text, XML
Performance	Graph Layout	Unordered	Ordered	Multiple	Unordered	Ordered
	Large Network	Difficult to interpret	Interpretable	Interpretable	Difficult to interpret	Interpretable

* The comparison is based on the information available from individual website as on Feb, 2011.