Visualization and analysis of protein interactions

Byong-Hyon Ju\textsuperscript{1}, Byungkyu Park\textsuperscript{1}, Jong H. Park\textsuperscript{2} and Kyungsook Han\textsuperscript{1,*}

\textsuperscript{1}Department of Computer Science & Engineering, Inha University, Inchon, 402-751, South Korea and \textsuperscript{2}MRC-DUNN, Human Nutrition Unit, Cambridge CB2 2XY, UK

Received on May 31, 2002; revised on July 23, 2002; accepted on August 14, 2002

ABSTRACT

Summary: We have developed a new program called InterViewer for drawing large-scale protein interaction networks in three-dimensional space. Unique features of InterViewer include (1) it is much faster than other recent implementations of drawing algorithms; (2) it can be used not only for visualizing protein interactions but also for analyzing them interactively; and (3) it provides an integrated framework for querying protein interaction databases and directly visualizes the query results.

Availability: http://wilab.inha.ac.kr/protein/

Contact: khan@inha.ac.kr

INTRODUCTION

Recent improvements in high-throughput proteomics techniques such as the yeast two-hybrid system (Uetz et al., 2000) have produced a rapidly expanding volume of protein interaction data. Protein interaction data can be visualized as a graph in which nodes represent proteins and edges represent their interactions. Most protein interaction data have the following characteristics: (1) When visualized as a graph, the data yields a disconnected graph with many connected components. (2) The data yields a non-planar graph with a large number of edge crossings that cannot be removed in a two-dimensional drawing. (3) The number of interacting proteins varies widely within the same set of data, resulting in nodes of very high degree as well as very low degree of interaction. (4) The data often contains protein interactions corresponding to self-loops.

Therefore, interaction data demands robust and diverse features in visualizing and analyzing complex information. This paper describes an algorithm and its implementation called InterViewer. Its integrated approach to databases can handle rapidly increasing diverse interaction data effectively compared to conventional flat files. InterViewer can directly query databases and visualizes the results in three-dimensional space at runtime. Visualized networks can be further refined or navigated to explore protein interactions.

ALGORITHM AND IMPLEMENTATION

InterViewer’s layout is based on the force-directed layout of Walshaw’s algorithm (Walshaw, 2000), but different from Walshaw’s algorithm in the following sense: (1) Walshaw’s algorithm groups nodes into clusters, whereas InterViewer does not. (2) Walshaw’s algorithm initially places nodes randomly, whereas InterViewer places nodes on the surface of a sphere for better results. (3) Walshaw’s algorithm iteratively updates layouts until the graph size falls below a certain threshold value, whereas InterViewer iterates 20 times unless specified otherwise by a user. At each iteration, the node positions are updated based on global spring forces between nonadjacent nodes (line 8 in Algorithm Layout) as well as local spring forces between adjacent nodes (line 10).

Algorithm 1 Layout

\begin{verbatim}
1: \( r = 1 \)
2: repeat
3: \( g = 0.01 \cdot r \cdot k^2 \) \hspace{1cm} \{k: natural spring length\}
4: for \( v \in V \) do
5: \( D = 0 \) \hspace{1cm} \{D: displacement vector of node \( v \)\}
6: for \( u \in V, u \neq v \) do
7: \( \Delta = \text{pos}[u] - \text{pos}[v] \) \hspace{1cm} \{\text{pos}[u]: position of node \( u \)\}
8: \( D = D - g \cdot (\Delta/|\Delta|) \cdot |u|/|\Delta| \)
9: \{\|u\|: distance of \( u \) from the origin\}
10: if \( u \in \Gamma(v) \) then
11: \( D = D - \Delta/|\Delta| \cdot (1 - |\Delta|/k)/|\Gamma(v)| \)
12: \{\|\Gamma(v)\|: set of vertices adjacent to \( v \)\}
13: end if
14: end for
15: \( \text{pos}[v_{new}] = \text{pos}[v_{old}] + D \)
16: end for
17: \( r = 0.98 \cdot r \)
18: until \( T \) times \[T: user-specified number or 20\]
\end{verbatim}

Let \( T \) be the total number of iterations of the outer loop (line 2 of Algorithm Layout). For a graph with \( n \) nodes, \( O(n) \) time is required to compute the displacement \( D \) of...
Table 1. Running times of graph drawing programs on 3 test cases on a
Pentium IV 1.7 GHz processor

<table>
<thead>
<tr>
<th>program</th>
<th>Y2H (1005 nodes, 905 edges)</th>
<th>MIPS-G (888 nodes, 1093 edges)</th>
<th>MIPS-P (2167 nodes, 2948 edges)</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterViewer</td>
<td>5 s</td>
<td>4 s</td>
<td>23 s</td>
</tr>
<tr>
<td>Pajek (F–R)</td>
<td>3 min 17 s</td>
<td>1 min 48 s</td>
<td>12 min 42 s</td>
</tr>
<tr>
<td>Tulip (GEM)</td>
<td>26 s</td>
<td>19 s</td>
<td>27 min 0 s</td>
</tr>
<tr>
<td>Tulip (S–E)</td>
<td>3 min 40 s</td>
<td>3 min 43 s</td>
<td>95 min 21 s</td>
</tr>
</tbody>
</table>

F-R: Fruchterman–Reingold’s layout, S-E: Spring-Electrical Force layout,
MIPS-G: MIPS genetic interaction data, MIPS-P: MIPS physical interaction data

a node, and $O(n^2)$ time is required to compute $D$ of all nodes. Therefore, the total time required is $O(T \cdot n^2) = O(n^2)$ since $T$ is constant. Compared to the asymptotic time complexity $O(n^3)$ of Kamada and Kawai’s algorithm (1989), InterViewer has a lower order of time complexity (see below for actual running times).

The layout algorithm was implemented in Borland Delphi 6.0, and databases of protein–protein interactions were constructed using Microsoft Data Access Components 2.7. The program runs on any PC with Windows 2000/XP/Me/98/NT 4.0.

Figure 1 shows the drawing of the entire MIPS physical interaction data. The drawing appears to have edge crossings, but it actually contains no edge crossing in the three-dimensional drawing. InterViewer allows the user to interactively explore three-dimensional drawings by rotating or by zooming in or out of them. It also enables the user to extract connected components of a disconnected graph, proteins interacting with a certain protein within a specified distance level, or proteins sharing a certain function. A protein interaction network can be saved either in an image file, the local database or a text file in GML format (http://www.uni-passau.de/Graphlet/GML).

For the purpose of comparison of actual running times, we ran two other graph-drawing programs, Pajek (Batagelj and Mrvar, 2001) and Tulip (David, 2001). Table 1 shows the running times of InterViewer, Pajek, and Tulip on a same set of test cases. It follows from this result that InterViewer is an order of magnitude faster than Pajek (Fruchterman–Reingold layout) and Tulip with Spring-Electrical Force layout and is significantly faster than Tulip with GEM layout. We also implemented Kamada and Kawai’s algorithm to compare its actual running times with ours. Kamada and Kawai’s algorithm produces 2D drawings only, so we extended it to 3D drawings. Since their algorithm cannot visualize a disconnected graph, we tested both their algorithm and ours on the largest connected components of Y2H data (473 nodes), MIPS genetic interaction data (531 nodes), and MIPS physical interaction data (1526 nodes) on a Pentium IV 1.7 GHz processor. The running times of Kamada and Kawai’s algorithm on these test cases are 3.7 s, 4.9 s, and 1 min 43 s, respectively, while those of InterViewer are 1.2 s, 1.5 s, and 12.7 s, respectively. Thus, InterViewer is faster than Kamada and Kawai’s algorithm in actual running times, too.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Information and Communication of South Korea under grant number IMT2000-C3-4.

REFERENCES


Fig. 1. Drawing of the MIPS physical interaction data with 2167 nodes and 2948 edges. Node labels are not shown in this drawing.