Eurographics/ IEEE-VGTC Symposium on Visualization (2007)
Ken Museth, Torsten Möller, and Anders Ynnerman (Editors)

Path Visualization for Adjacency Matrices

Zeqian Shen[†] and Kwan-Liu Ma[‡]

Visualization and Interaction Design Institute
University of California at Davis

Abstract

For displaying a dense graph, an adjacency matrix is superior than a node-link diagram because it is more compact and free of visual clutter. A node-link diagram, however, is far better for the task of path finding because a path can be easily traced by following the corresponding links, provided that the links are not heavily crossed or tangled. We augment adjacency matrices with path visualization and associated interaction techniques to facilitate path finding. Our design is visually pleasing, and also effectively displays multiple paths based on the design commonly found in metro maps. We illustrate and assess the key aspects of our design with the results obtained from two case studies and an informal user study.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques; H.5 [Information Systems]: Information Interfaces and Presentation; G.2.2 [Graph Theory]: Path and Circuit Problems

1. Introduction

Graphs are common information structures for modeling relations between discrete entities. Many real world systems take the form of graphs. For example, a social network represents relationships between actors (such as individuals, families, Internet social groups, corporate organizations, business partners, nations, etc). Analyzing a social network provides structural intuition according to the ties linking actors. This leads to improved understanding of social processes.

Graphs are inherently visual. Yet, traditional node-link diagrams of graphs, connecting the graph nodes with lines, do not scale as the graph size and density increase. For a dense graph in particular, the large number of edge crossings create great visual complexity. Automatic layout algorithms have been developed to reduce edge crossing as much as possible, but they have a high computational cost and heuristic nature. Thus, node-link diagrams are not always a good choice for displaying graphs. On the other hand, adjacency matrices offer a very compact representation of graphs. Vertices are placed along the horizontal and vertical axes in the same order. Therefore, no layout computing is needed.

In an adjacency matrix, a non-zero entry represents an edge between the two corresponding vertices in a graph. Because each edge is defined by itself in a non-shared space, there is no edge crossing problem. However, an adjacency matrix view of a graph has its limitations. According to previous user studies [GFC05, KEC06], adjacency matrices are clearly better than node-link diagrams for displaying graphs with twenty or more nodes which are largely connected. However, for the task of path finding, a node-link diagram allows the user to quickly locate the paths of interest by simply following the associated links in the picture. Finding a path in an adjacency matrix is not a trivial task, especially when the path is composed of many nodes.

In this paper, we show how we augment adjacency matrices with interactive path visualization to remove the intrinsic limitation of a matrix representation. Our design superimposes path information on top of the matrix. The paths are displayed in an unambiguous and visually pleasing fashion with minimum crossing. We also use an idea from metro map visualization to more neatly display multiple paths. We have tested our design using a variety of graphs and two case studies using a social network and a food web are presented

[†] e-mail: zqshen@ucdavis.edu
[‡] e-mail: ma@cs.ucdavis.edu

in this paper. We also report our findings from an informal user study to determine whether the graphical path augmentation is helpful or not.

2. Background

Adjacency matrices are widely used in graph visualization because they can effectively display a dense graph. Interpreting the structural information buried in a matrix view of a graph requires some practice. In this section, we first give a brief introduction to interpreting adjacency matrices, followed by an overview of related visualization approaches. Finally, we discuss the challenges presented by the path finding problem.

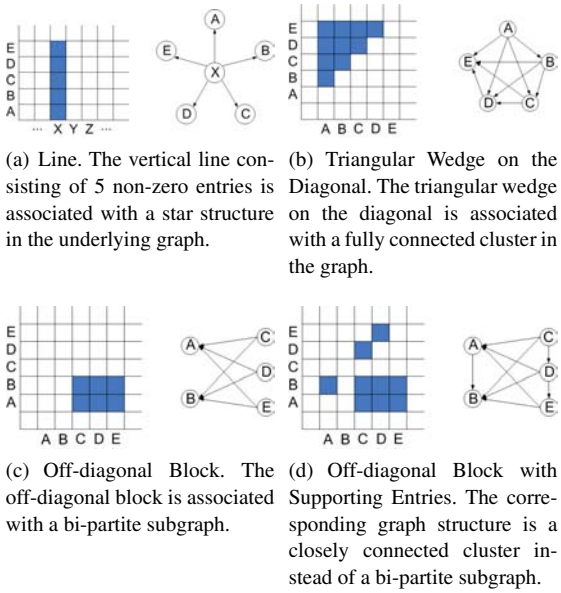
2.1. Adjacency Matrices

Adjacency matrices can be used to represent both directed and undirected graphs. For directed graphs, a non-zero entry at the intersection of column i and row j represents that there is a connection from vertex i to vertex j. For undirected graphs, the adjacency matrix is symmetric. Each entry on the matrix's diagonal represents a self-link of the corresponding vertex. Usually, self-links are ignored, thus entries along the diagonal are zero.

Patterns that consist of non-zero entries are key to interpreting adjacency matrices. Different patterns convey different structural properties of the underlying graph. For instance, horizontal or vertical lines might indicate a star structure in the graph. In Fig. 1(a), the vertical line consisting of 5 non-zero entries reveals that there are connections from vertex X to vertex A, B, C, D and E. Closely connected clusters are presented as triangular wedges on the diagonal (See Fig. 1(b)). Blocks away from the diagonal may indicate bi-partite subgraphs (See Fig. 1(c)). There are two groups of vertices and only inter-group edges. The discussions above assume that the patterns are stand-alone, because existence of supporting entries elsewhere will totally change the interpretation. Fig. 1(d) shows an example of an off-diagonal block with supporting entries.

Ordering of vertices on vertical and horizontal axes has a tremendous impact on the effectiveness of adjacency matrix visualization. A matrix with poor ordering is indistinguishable from noise while one with good ordering shows salient patterns. Comprehensive discussions of matrix ordering can be found at [CM69, Wil86, Kin70, MML07]. In this paper, breadth-first-search ordering is used for all examples.

Adjacency matrices are often used as overviews of large graphs because of their compactness. For instance, MatrixExplorer, designed for social network analysis, is equipped with both matrix and node-link views [HF06]. Users are allowed to investigate the global structure of a graph in the matrix view and choose subgraphs of interest for details in



(a) Line. The vertical line consisting of 5 non-zero entries is associated with a star structure in the underlying graph.

(b) Triangular Wedge on the Diagonal. The triangular wedge on the diagonal is associated with a fully connected cluster in the graph.

(c) Off-diagonal Block. The off-diagonal block is associated with a bi-partite subgraph.

(d) Off-diagonal Block with Supporting Entries. The corresponding graph structure is a closely connected cluster instead of a bi-partite subgraph.

Figure 1: *Patterns of Adjacency Matrices*

the node-link view. Moreover, an adjacency matrix has uniform visual representation, which makes it perfect for multiscale visualization. Frank van Ham introduces an interactive multilevel matrix visualization system for large hierarchical software call graphs [vH03]. MGV, a system for visualizing massive multidigraphs also utilizes this property [AK02]. User studies were conducted by both Ghoniem [GFC05] and Keller [KEC06], in which the readability of matrices and node-link diagrams were compared. Ghoniem focuses on the tasks that reveal graph topological features, while Keller focuses on tasks involving semantic information. Both studies concluded that adjacency matrices outperform node-link diagrams for large dense graphs, except for the task of path finding.

2.2. Path finding

Path-finding is the task of finding a series of continuous edges connecting two vertices. Finding the shortest path is the most common task. However, in some cases, we also need to find multiple paths between vertices. Path-finding is very important for graph analysis. In social network analysis, for example, connections between two individuals indicate the possibility that they will meet in the future.

Paths are relatively easy to recognize in node-link diagrams because they are represented as a series of connected line segments. In adjacency matrices, edges are represented as entries. Thus, a path is represented as a set of entries, in which the row index of each entry is the same as the column index of the next one; i.e., a path $P = \{v_0, v_1, v_2, ..., v_n\}$ of graph $G(V,E)$ is equivalent to

a series of entries in the corresponding adjacency matrix, $\{e[v_0,v_1],e[v_1,v_2],...,e[v_{n-1},v_n]\}$, where $e[v_i,v_j]$ is an entry at the intersection of column v_i and row v_j. Unlike clusters or star structures, a path can appear as any loose pattern in an adjacency matrix. Fig. 2 shows an example of a path in a node-link diagram and its equivalent adjacency matrix. To find a path in an adjacency matrix, the user needs to go back and forth between the columns and rows, which can be especially difficult for long paths or large graphs. Thus, our research aims at designing effective visualization highlighting paths in adjacency matrices. The visualization should allow the viewer to easily follow each path from beginning to end and obtain vertices along the path. To our knowledge, no such research has been done for this problem.

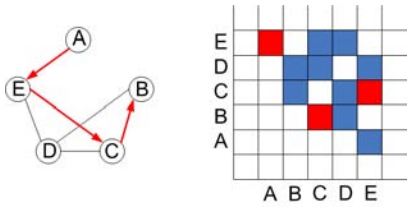


Figure 2: *Comparison of a Path in an Adjacency Matrix and the Equivalent Node-link Diagram. The path illustrated in this example is $A \rightarrow E \rightarrow C \rightarrow B$. In the adjacency matrix, the corresponding entries $\{e[A,E],e[E,C],e[C,B]\}$ are highlighted in red.*

3. Visualizing a Single Path

We first address the problem of visualizing a single path. Given two vertices, we choose to visualize the shortest path, which is usually the most critical path between them. In our interactive matrix visualization system, users can select a starting vertex from the vertical axis and an ending vertex from the horizontal axis. The shortest path is then computed using Dijkstra's algorithm [Dij59] and is visualized over the matrix. The rest of this section discusses related problems, including path layout, rendering, and how to eliminate ambiguities caused by path crossing. In the rest of the paper, a small social network of a group of students is used as the example to demonstrate our path visualization methods. It is an undirected graph with 25 vertices and 25 edges. The actual names are removed for privacy.

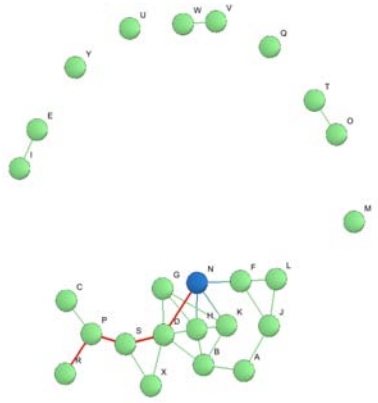
3.1. Path Layout

Visualizing the shortest path in node-link diagrams is straightforward. Simply highlighting all edges along the path is sufficient (See Fig. 3(a)). Following the highlighted edges, users can easily find the vertices and edges on the path. In adjacency matrix, a path $P = \{v_0,v_1,v_2...v_n\}$ is equivalent to a series of entries, $\{e[v_0,v_1],e[v_1,v_2],...,e[v_{n-1},v_n]\}$.The column index associated with the first entry indicates the starting vertex of the path, and the row indices associated with all entries indicate the rest of the vertices on the path. Thus, we choose to draw a vertical dotted line between the first entry and the horizontal axis and a horizontal dotted line between each entry and the vertical axis. With the help of dotted lines, users can easily find the vertices on the path. We also connect the entries, such that users can follow the path. In Fig. 3(b), the same path highlighted in Fig. 3(a) is visualized using this method. Although the dotted lines provide a clear guide to finding corresponding vertices, it is still difficult for users to go back and forth between the axes and entries.

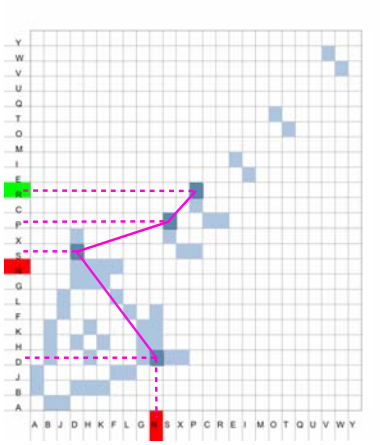
One fact about adjacency matrices is often ignored. Besides columns and rows, the diagonal can also represent vertices. Since ordering algorithms make the non-zero entries gather along the diagonal, it is much easier to use the diagonal as the reference for vertices than the axes. Thus, we use horizontal lines to connect the entries and the diagonal. For an entry $e[v_{i-1},v_i]$ on the path, $P = \{v_0,v_1,v_2,...,v_n\}$, the connected entry on the diagonal is $e[v_i,v_i]$. The next entry on the path is $e[v_i,v_{i+1}]$, which can be connected to $e[v_i,v_i]$ by a vertical line. Repeatedly doing this, we can connect all the entries related to vertices and edges on the path. Fig. 3(c) illustrates this new design. In this example, the path $P = \{N,D,S,P,R\}$ is visualized as a series of lines connecting entries $e[N,N],e[N,D],e[D,D],e[D,S],e[S,S],e[S,P],e[P,P],e[P,R]$ and $e[R,R]$. The series of lines carry all information of the original path and agree with the underlying matrix-based representation. To make the visualization clearer, the entries associated with edges are highlighted, and those associated with vertices are labeled.

3.2. Path Crossing

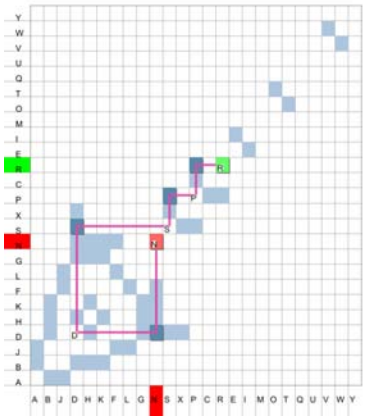
One of the major problems for our design is path crossing (See Fig. 4(a)). Limited by the given space, some of the figures in this paper do not display the complete adjacency matrix. Because lines are either vertical or horizontal, ambiguities are created at the intersections. In order to solve the problem, we experimented with three different designs. First, the crossing annotation used in an electric circuit diagram is applied (See Fig. 4(b)). At each intersection, one of the lines are drawn as an overpassing curve. The second method draws the lines with white boundaries to enhance the boundary perception (See Fig. 4(c)). The third design uses curves instead of lines, because ambiguities are caused by the regular layout of lines. The two lines connecting three consecutive entries, $e[v_i,v_i],e[v_i,v_j]$ and $e[v_j,v_j]$, are replaced by a cubic B-spline. The three entries are used as control points for the curve (See Fig. 4(d)). Ambiguities at intersections are eliminated because the path crossings are no longer orthogonal. We presented the three designs to a group of users. Most of them liked the design using curves because it is visually more pleasing. However, some users who had advanced knowledge of adjacency matrices and graphs, argued that the curves lose critical information,



(a) Path $\{N, D, S, P, R\}$ is Highlighted in the Node-link Diagram. The layout is computed using a force-directed method [KK89]

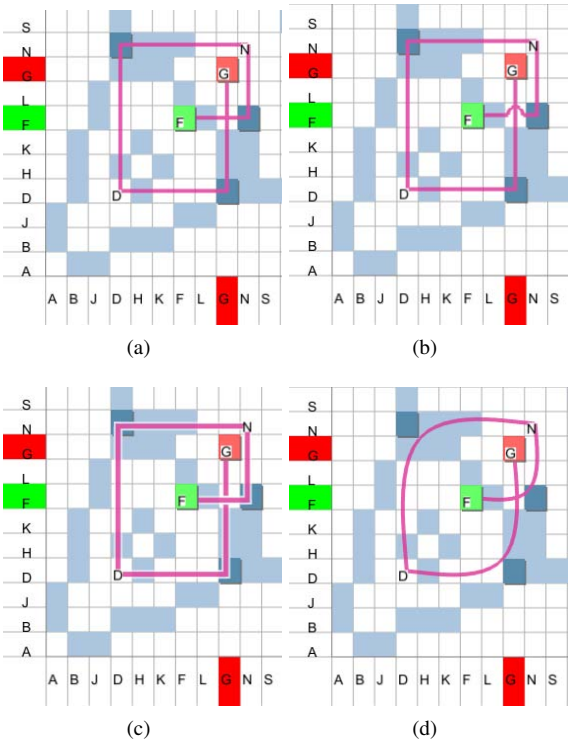


(b) The First Approach: Connecting matrix entries and the axes. Column and row indices associated with the vertices on the path are highlighted by dotted lines.



(c) The Second Approach: Connecting matrix entries and the diagonal. Vertices on the path are the labeled entries on the diagonal.

Figure 3: *Path Layout in Adjacency Matrices*



(a) (b) (c) (d)

Figure 4: *Path Crossing Problem. In (a), ambiguities exist at the intersections because of the regular layout of the paths. Three designs are proposed here. (b) Draw one of the lines at an intersection as an overpassing curve. (c) Draw lines with white boundaries to enhance the boundary perception at intersections. (d) Draw cubic B-splines instead of lines. Ambiguities are eliminated because intersections of curves are nonorthogonal.*

because the curves do not touch the entries associated with edges on the path. Between the designs of overpassing and enhanced boundary, users had various opinions. Thus, all three methods are provided as options in our system and the one using curves is set to be the default.

3.3. Rendering

For directed graphs, conveying the direction of each path is important. We use colors to indicate the direction. The entries change from red to green along the path (See Fig. 5(a)). Opacity is another method to indicate the direction. Fig. 5(b) shows the results of opacity decreasing along the path. In this image, rows are also colored, and the opacity is negatively correlated to the distances from the associated vertices to the starting vertex. This functionality enables users to quickly find vertices within the vicinity of a selected vertex.

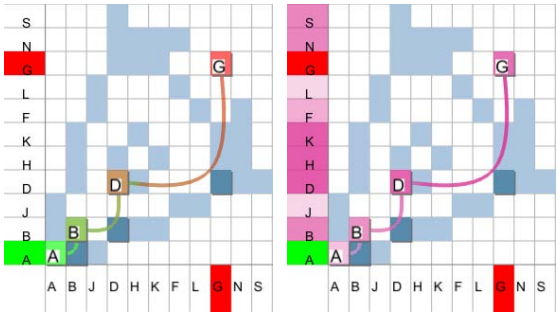


Figure 5: *Rendering Options for Path Visualization. (a) Color is used to indicate path direction. (b) Opacity is used to indicate path direction. The rows are colored based on the distances from their associated vertices to the starting vertex.*

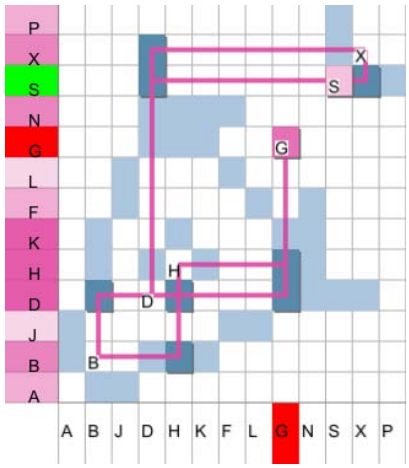


Figure 6: *Visualization of Multiple Paths Using the Approach for A Single Path. Ambiguities caused by path overlapping make the visualization unreadable.*

4. Visualizing Multiple Paths

Besides the shortest path, other paths might also be interesting. A new design for visualizing multiple paths is presented in this section. We borrow the visual design of metro maps and develop path routing algorithms for reducing path crossing.

4.1. Path Filtering

The first problem we face is the large number of paths between two vertices for dense graphs. In a fully connected graph with 10 vertices, there are $2^{(10-2)} = 2^8 = 256$ paths between any pair of vertices. For large graphs, not only is visualizing all the paths infeasible, but computing all the paths is also expensive. Furthermore, users often only care about the n shortest paths or paths shorter than a certain length, because shorter paths usually indicate more important connections between entities. Our system allows users to specify the maximum path length l and the maximum number of visible paths n. The paths longer than l are dropped during path searching to save computing time, and then the n shortest paths are visualized.

4.2. Path Overlapping

Approaches for a single path are applied on multiple path visualizations. Fig. 6 shows the visualization of the 5 shortest paths from vertex G to vertex S. Although there are only 5 paths, it is not easy to follow each path in the visualization. The major problem is the ambiguities caused by path overlapping. Many paths share the same lines, and thus obscure each other. This problem is very similar to what metro map drawing deals with, where different metro routes share the same tracks. Fig. 7 shows a part of the metro map of Koln, Germany (http://www.vrsinfo.de/). Metro routes are

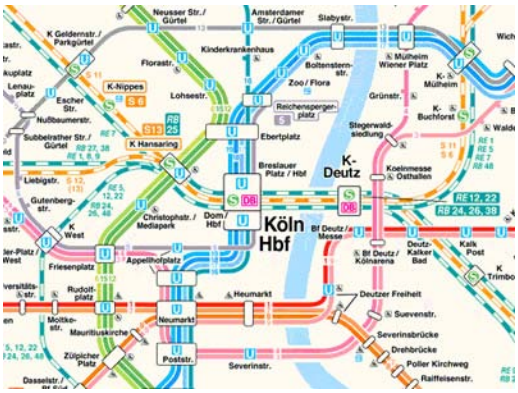
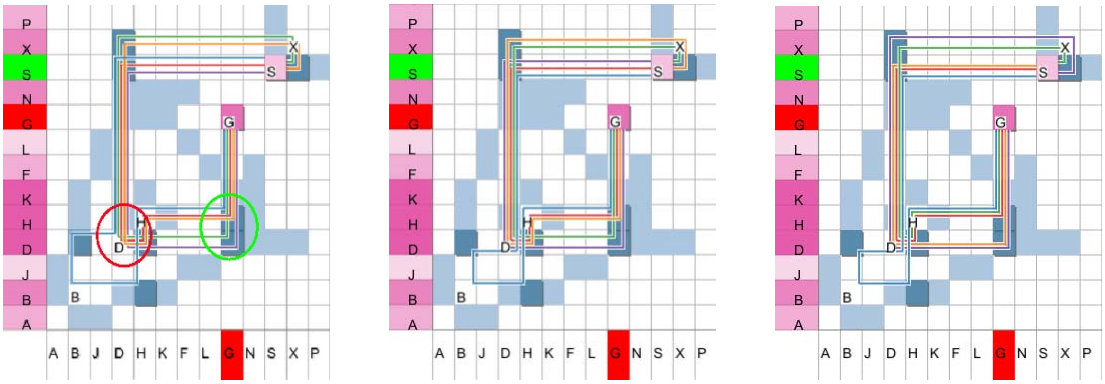


Figure 7: *A Part of the Metro Map of Koln, Germany.*

laid in parallel to avoid overlapping, and different colors are assigned to the routes. The routes are drawn either vertically or horizontally or at 45° to increase readability. Lines are drawn with white boundaries to enhance separations between parallel lines and also to resolve ambiguities at intersections. Fig. 8(a) shows the visualization of the same paths in Fig. 6 following these principles. All the paths are clearly presented and can be easily followed.

4.3. Path Crossing

Large numbers of path crossings affect the readability of the parallel layout design. The areas circled in Fig. 8(a) contain two typical types of crossings. The one in the red circle contains crossings happening at turn points. These kind of crossings can be eliminated by adjusting turning order of parallel paths. At each corner, the inner paths should turn



(a) Parallel Layout of Paths. The area in the red circle contains crossing can be reduce by adjusting the turning order of paths. The crossings in the green circle are caused by improper placement of paths.

(b) Crossings at corners are eliminated by adjusting turning order of paths.

(c) Crossings are further reduced by adjusting the order of paths based on the turns they make.

Figure 8: *Multiple Path Visualization*

earlier than the outer ones (See Fig. 8(b)). The crossings in the green circle can be reduced by proper ordering of paths. We developed an algorithm to arrange the paths based on the turns they make. When we travel down along the paths, the paths turning earlier should be the outer ones, while ones that turn later should be put in the middle. Paths turning left are put on the left side, and ones turning right on the right side. Fig. 8(c) shows the result after adjusting path ordering using our algorithm. Most crossings are eliminated and visual clutters are thus reduced. Moreover, similar paths are bundled together, which makes them much easier to follow.

5. Results

Two case studies are presented here to demonstrate the capability of our path visualization design.

5.1. Social Network

In this case study, the friendship network of a small hi-tech computer firm is used [Kra99]. The network is an undirected graph, which contains 36 vertices and 147 edges. Using our design, we can easily find the shortest connection between two individuals (See Fig. 9(a)). By setting the maximum path length to 2, we are able to find all the common neighbors of two individuals (See Fig. 9(b)).

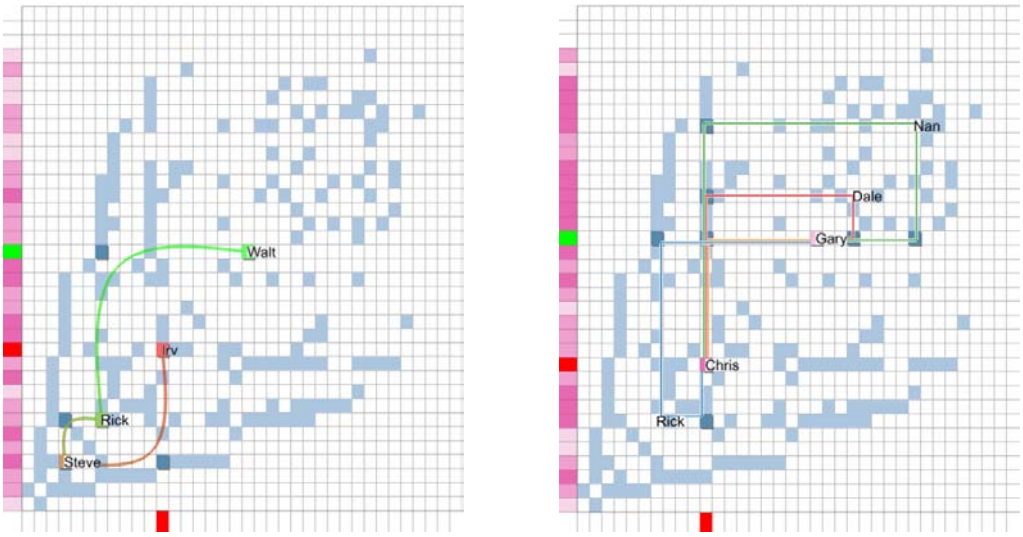
5.2. Food Webs

Food webs describe feeding relationships between species in a biotic community. They are usually represented as directed graphs with vertices for species and edges for feeding relationships. An edge $e(v_i, v_j)$ indicates that species v_j feed upon species v_i. A food chain is a single path in the food web, which reveals how the energy and material may flow in the ecosystem. Fig. 10 shows a food chain in the food web describing the ecosystem at Chesapeake Bay [BU89]. The web contains 39 species and 177 feeding relationships. The food chain starts with zooplankton, followed by bay anchovy, sediment particulate organic, bacteria in sediment POC, other polychaetes, and catfish. It shows how energy flows from zooplankton to catfish through some intermediate species.

Each food web contains a Input entity and Output entity, which are the ports to exchange material and energy with the surrounding environment. The energy enters the ecosystem from Input and leaves the system at Output. Thus, paths from Input to Output are energy flows that go through the food web. We compare the 5 shortest paths from Input to Output in the food web of Chesapeake Bay with the food web of Lake Michigan [BU89] (See Fig. 11). The energy flows are very different. In Fig. 11(a), the energy from Input are retrieved by phytoplankton and passed through various species to the Output. In Fig. 11(b), the energy are retrieved by various species instead of a single one. The common path to the Output is through Detritus. Thus, the ecosystem of Chesapeake Bay and Lake Michigan is very different.

6. User Feedback

Informal user studies are conducted to demonstrate our path visualization design. Most of the participants are our fellow researchers, PhD students. They all have experience with information visualization, and some have advanced knowledge

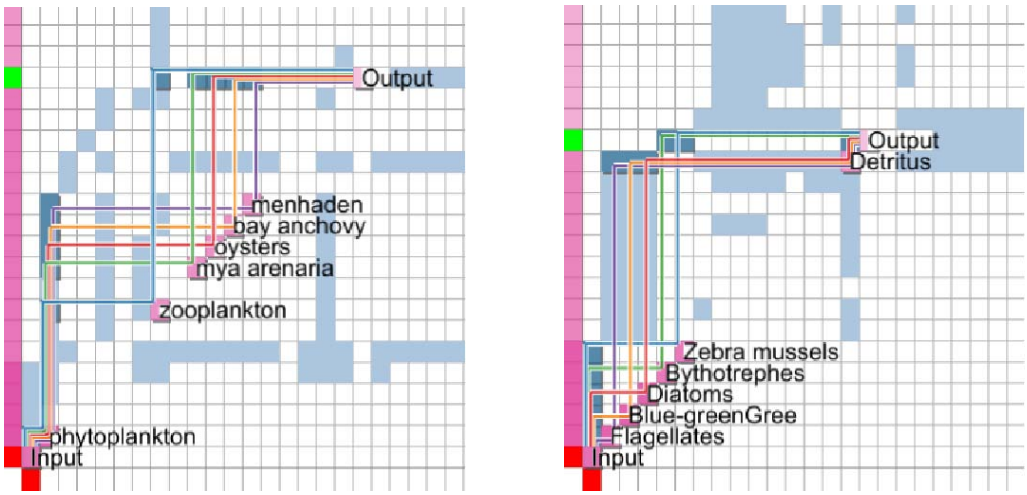


(a) The shortest connection between Irv and Walt goes through Steve and Rick.

(b) All the connections of length 2 between Chris and Gary. The intermediate vertices on the path are actually the common neighbors of Chris and Gary.

Figure 9: *Friendship Network of a Hi-Tech Firm.*

.



(a) Energy Flow From Input to Output in Food Web of Chesapeake Bay.

(b) Energy Flow From Input to Output in Food Web of Lake Michigan.

Figure 11: *Comparison of the Energy Flow in Food Webs of Chesapeake Bay and Lake Michigan.*

of graph and matrix visualization. A few participants are researchers in areas of social science or complex networks. The majority of the participants regarded the technique as useful for easily finding paths in adjacency matrices. Those participants who have experience with matrix visualization found that our design significantly enhances the path finding capability of adjacency matrices. The visualization was also regarded as being aesthetically pleasing, especially the curve-path and the parallel-path visualization designs. In addition, some participants pointed out that interactive path visualization helped them better understand the adjacency matrices.

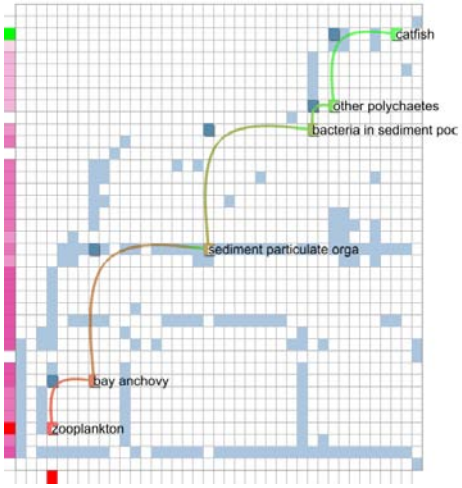


Figure 10: *Food Web of Chesapeake Bay. The highlighted path is a food chain that starts from zooplankton and ends at catfish. It indicates the a series of feeding relationships and also an energy flow in the ecosystem.*

7. Conclusions

We have introduced an effective design for path visualization in adjacency matrices. A path is visualized as a series of lines connecting the matrix entries associated with edges and vertices on the path. Overpassing, boundary enhancement and curves are used to eliminate ambiguities at path intersections. We have also addressed the problem of visualizing multiple paths. Using an idea from metro maps drawings, overlapping paths are drawn as parallel paths. Path routing algorithms are developed to reduce path crossing. In the user study, the participants particularly valued the neat visualization of parallel paths.

The scalability of our design remains to be studied. The visibility of paths relies on the number of pixels per cell entry in the matrix. In addition, the parallel line visualization for multiple paths suffers visual complexity when there are too many paths to display. Lensing techniques could solve the problem. Another point of interest for future work would be using animation to show path direction.

8. Acknowledgment

This research was supported in part by the U.S. National Science Foundation through grants CCF-0634913, IIS-0552334, CNS-0551727, OCI-0325934, CCF-9983641, and CCF-0222991, and the U.S. Department of Energy through the SciDAC program with Agreement No. DE-FC02-06ER25777, DOE-FC02-01ER41202, and DOE-FG02-05ER54817. We would also like to thank the reviewers and the members of VIDI for their constructive comments and suggestions.

References

[AK02] ABELLO J., KORN J.: Mgv: A system for visualizing massive multidigraphs. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 21–38.

[BU89] BAIRD D., ULANOWICZ R.: The seasonal dynamics of the chesapeake bay ecosystem. *Ecol. Monogr. 59* (1989), 329–364.

[CM69] CUTHILL E., MCKEE J.: Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 24th national conference* (New York, NY, USA, 1969), ACM Press, pp. 157–172.

[Dij59] DIJKSTRA E. W.: A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269âĂŞ–271.

[GFC05] GHONIEM M., FEKETE J., CASTAGLIOLA P.: On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization 4*, 2 (2005), 114–135.

[HF06] HENRY N., FEKETE J.-D.: Matrixexplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 677–684.

[KEC06] KELLER R., ECKERT C. M., CLARKSON P. J.: Matrices or node-link diagrams: which visual representation is better for visualising connectivity models? *Information Visualization 5* (2006), 62–76.

[Kin70] KING I.: An automatic reordering scheme for simultaneous equations derived from network systems. In *Int. J. Numer. Meth. Eng.* (1970), pp. 479–509.

[KK89] KAMADA T., KAWAI S.: An algorithm for drawing general undirected graphs. *Inf. Process. Lett. 31*, 1 (1989), 7–15.

[Kra99] KRACKHARDT D.: The ties that torture: Simmelian tie analysis in organizations. *Research in the Sociology of Organizations 16* (1999), 183–210.

[MML07] MUELLER C., MARTIN B., LUMSDAINE A.: A comparison of vertex ordering algorithms for large graph visualization. In *Proc. Asia-Pacific Symposium on Visualisation 2007* (2007).

[vH03] VAN HAM F.: Using multilevel call matrices in large software projects. In *Proc. of the IEEE Symposium on Information Visualization (INFOVIS'03)* (2003).

[Wil86] WILSON R.: *Introduction to graph theory*. John Wiley & Sons, Inc., New York, NY, USA, 1986.