SAND REPORT
SAND2002-1533
Unlimited Release
Printed June 2002

Six Methods of Transaction Visualization in Virtual Environments

Eleanor A. Walther and Michael W. Trahan
Systems Technology Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1207

Kenneth L. Summers
Department of Computer Science
Timothy Eyring and Thomas P. Caudell
Department of Electrical and Computer Engineering
University of New Mexico
&
Albuquerque High-Performance Computing Center

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/ordering.htm

SAND2002-1533
Unlimited Release
Printed April 2002

Six Methods of Transaction Visualization in Virtual Environments

Eleanor A Walther and Michael W. Trahan
System Technologies Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-1207

Kenneth L. Summers
Department of Computer Science
Timothy Eyring and Thomas P. Caudell
Department of Electrical and Computer Engineering
University of New Mexico
&
The Albuquerque High-Performance Computing Center
Albuquerque, New Mexico

Abstract

Many governmental and corporate organizations are interested in tracking materials and/or information through a network. Often, as in the case of the U.S. Customs Service, the traffic is recorded as transactions through a large number of checkpoints with a correspondingly complex network. These networks will contain large numbers of uninteresting transactions that act as noise to conceal the chains of transactions of interest, such as drug trafficking. We are interested in finding significant paths in transaction data containing high noise levels, which tend to make traditional graph visualization methods complex and hard to understand. This paper covers the evolution of a series of graphing methods designed to assist in this search for paths—from 1-D to 2-D to 3-D and beyond.

This Page Intentionally
Left Blank

Acknowledgements

This work was funded by Sandia National Laboratories' Laboratory Directed Research and Development (LDRD) funds. Additional thanks goes to Steve Smith of Los Alamos National Laboratory for suggesting the name for the Dreamcatcher.

This Page Intentionally
Left Blank

Contents

Figures

Nomenclature

AHPCC	Albuquerque High-Performance Computing Center
HUD	heads-up display
IDL	Interactive Data Language
3-D	three-dimensional
2-D	two-dimensional
UNM	The University of New Mexico
VE	virtual environment
LDRD	Laboratory Directed Research and Development

Introduction

The U.S. Customs Service has tried to increase the chances of intercepting illicit drugs at the border points of entry by entering the transportation processes upstream. Starting in 1998, the Customs Service initiated the Americas Counter Smuggling Initiative, which sent Customs agents throughout Latin America to develop systems to track shipping and transportation systems starting in foreign ports [1].

The shipping systems originate outside the United States and enter through various ports of entry. This movement of commodities gives rise to a large network. This network can be represented as a graph where one is interested in finding the paths associated with illicit drug traffic. These paths potentially provide information about causal relationships between the nodes. The nodes of the graph are points where shipping containers are (or could be) inspected, and the arcs of the graph are the flow of the material between these checkpoints. The transit time between these points is a variable and can be up to 30 days. The abstraction of this problem is to find paths or patterns in a large graph. Successive links in the path must have increasing start times associated with them. We make the simplifying assumption that the commodity is not subdivided, but stays as a unit until it reaches its destination. This type of analysis is known as link analysis.

A data set of this type is large and contains more noise than data of interest. Noise, in this case, consists of normal transactions that occur naturally, but do not contribute to the types of paths we are interested in. Our approach to analyzing this data is to develop 3-D visualization techniques that highlight the time dependencies in order to decrease the number of potential paths of interest. We chose to visualize this data by mapping it as a graph and using graph-visualization techniques.

A random 2-D graph is the obvious first step. It would be possible to arrange the nodes (geographic locations) so that there are minimal crossing of edges (transactions), but in a large, highly connected network the minimal number of edge crossings is likely to be large. Graphs with large number of edge crossing contribute to confusion when one tries to trace paths of interest. The data does not lend itself to hierarchical representation, thereby eliminating a large number of graph-visualization techniques such as trees [2], cone trees [3, 4], or semantic zooming techniques [5]. What we are looking for involves paths through a block of data, so representations that impose order on the data seem like a reasonable approach.

The six methods for visualizing this data discussed in this report are outlined below. These techniques are based on circle graphs, new and innovative use of parallel-axes graphs [6], and a new technique for folding adjacency matrices.

- Method 1 and raw data,
- Method 2 and circle graph,
- Method 3 and 2-D adjacency matrices in three axes,
- Method 4 and 3-D adjacency matrix,
- Method 5 and Parallel axes graph, and

- Method 6 and Dreamcatcher (expanded adjacency matrix).

Link analysis is the process of mapping connections between entities. The purpose of link analysis is to mine the connection data so as to discover previously unexpected paths between seemingly unrelated entities. Connections may represent many things: sales, shipments, physical contacts or relations, circles of influence, etc. Connections may be time dependent, and duration or volume dependent, etc. Link analysis is used in many fields including market research, law enforcement, epidemiology, and fraud detection.

Linkage data typically can be modeled as a graph, with nodes representing entities of interest and links (also known as arcs) representing relationships or transactions. Link analysis attempts to infer useful knowledge from a large body of textual information. Our approach is to represent entities of interest as nodes in a graph and the relationship or transactions as arcs in a graph.

Connections between nodes form paths of one or more links. For example, we may be interested in identifying all paths no more than "N" links long from a given starting entity (known as the source node) to a given ending entity (called the destination node). As "N" increases, the number of possible paths increases correspondingly, making analysis of the data more complicated. One may be interested in paths between specified entities that occur often or in paths that are out of the ordinary.

Visualization of paths between nodes, especially multi-link paths, is difficult.

Link analysis may also attempt to answer such questions as:

- Is there a path between selected nodes?
- Which nodes are central to the network?
- Which links can be disrupted to most effectively impede the network's operation?

Current Visualization Methods

Currently, most link-analysis-visualization methods consist of a simple connected graph or a connection matrix.

The connected graph (Figure 1) shows links between nodes. Line width or colored edges can convey additional information about the connection, such as transaction type, frequency or volume of the transaction, or the transaction's duration. The connected graph usually is drawn to minimize the number of arc crossings. Some packages allow the user to drag and drop nodes so the user can improve the graph's layout interactively.

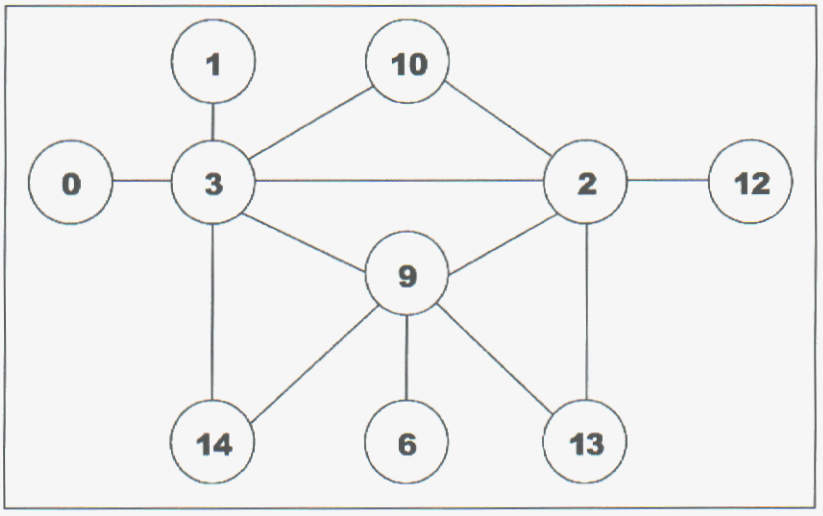

Figure 1. A connection graph.

There are several problems with the connected graph as a visualization method for link analysis. First, the graph does not display a time element, making it impossible to determine the order of transactions. Second, there is no way to distinguish the direction of transactions ("A to B" is not the same as "B to A"). Third, multi-link paths are not discernable.

The connection matrix (Figure 2) shows what nodes were connected at a particular time or during a specific time period. Again, the use of color can provide additional information, such as type of transaction, duration of transaction, etc. Unlike the connected graph, the direction of transactions is implicit in the connection matrix.

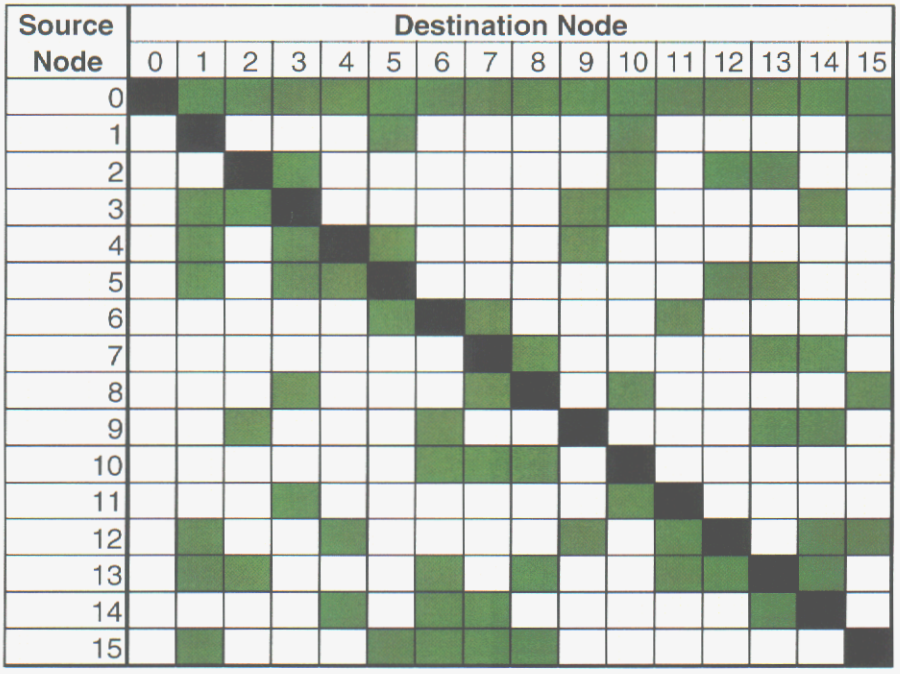

Figure 2. A connection matrix.
Green blocks mark active connections, and black blocks show impossible connections.

There are several problems with using the connection matrix as a visualization method for link analysis. First, although there is an implicit time element, the user must examine several connection matrices to determine the order and duration of transactions. Second, multi-link paths are still not distinguishable.

Visualizing Transactions

The Data

We have assumed that the data that we are displaying is the data returned from a database query. For example, a query might be "find all transactions originating from a specified entity within a certain period of time." The result of such a query could be a small number of nodes (on the order of 4 to 32), with hundreds of transactions over the queried time period, or it could be a large number of nodes and subsequent transactions.

For purposes of discussion here we will be using an example dataset with 243 transactions between 16 nodes covering 256 time steps. All values are integers.

The 2-D Tool

Our first visualization tool (Figure 3) was developed with a commercial package called Interactive Data Language (IDL) by Research Systems Inc. of Boulder, Colo. IDL was selected because it runs on a wide range of platforms and operating systems and provides a rich set of graphics and user interface tools and because project members were familiar with and have used IDL. This visualization tool supports a number of alternative views of the data and uses color to enhance the user's understanding of the data.

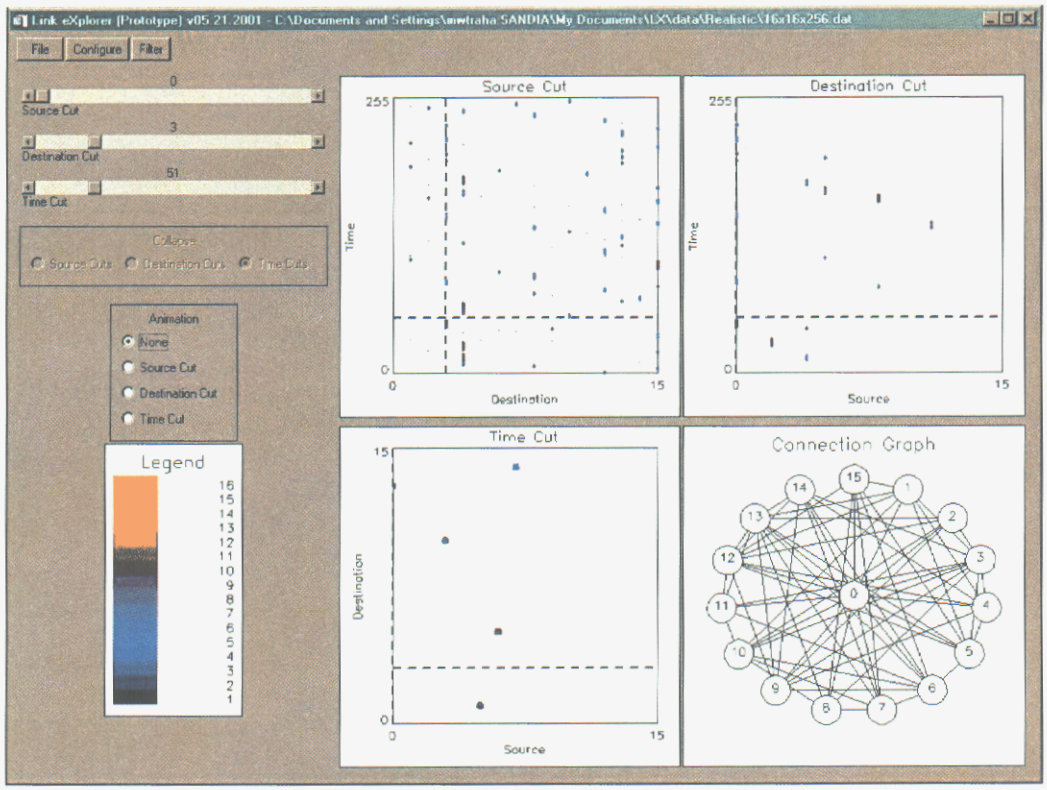

Figure 3. The 2-D visualization tool.

There are four primary display windows. The upper left window is a source cut. The source is the starting entity. In this example our source is node 0. This display shows what nodes are connected (by one link) to 0 and at what time step. The upper right window is a destination cut. The destination cut displays all the links coming into the destination node and at what time step. In this example the destination node is entity 3.The lower left window is a time cut. This display shows which nodes are connected at a given time step by a single connection. This is the traditional method of displaying a connection matrix. All three cuts can be animated. Color in the cuts represents the duration of a transaction, its type, or any other data of interest associated with the transaction. The legend illustrates the relative meanings of colors in the cuts. The dashed lines in the cut windows shows the positions of the other cuts as an aid in user orientation. The lower right window shows a simple-connection graph.

Four other views of the data can replace the lower right window. The collapsed time cut (Figure 4) is an alternate representation of the connection graph. It is easier to identify the 1-link transactions between entities in this representation than it was in the circular connection graph because there are no arc crossings. However, it is still not possible to construct possible paths of interest, that is, transactions between successive entities that must occur in some time sequence.

The collapsed source graph (Figure 4) shows which entity and at what time step the specified source entity is connected to other entities. In this view, the source initiates the transactions. This graph is useful in determining transactions that occur frequently between two given entities, may occur infrequently, or may occur with some time regularity.

The collapsed destination graph (Figure 5) is similar to the collapsed source graph in that it also shows a given entity and at which time step it is connected to other entities. For this view other entities initiate transactions, and the destination entity receives the transactions. Finally, the user can use this window to display links of interest over time (Figure 5). For example, the user can enter pairs of entities. The pairs are displayed on the vertical axis, and time is displayed on the horizontal axis. The pairs can be specified so that successive links form a potential path of interest. In this example, the path of interest is (0:3) (3:9) (9:2). If recurrent patterns exist, this graph presents the information with much of the noise filtered out. However, the analyst needs to be able to specify the path in advance.

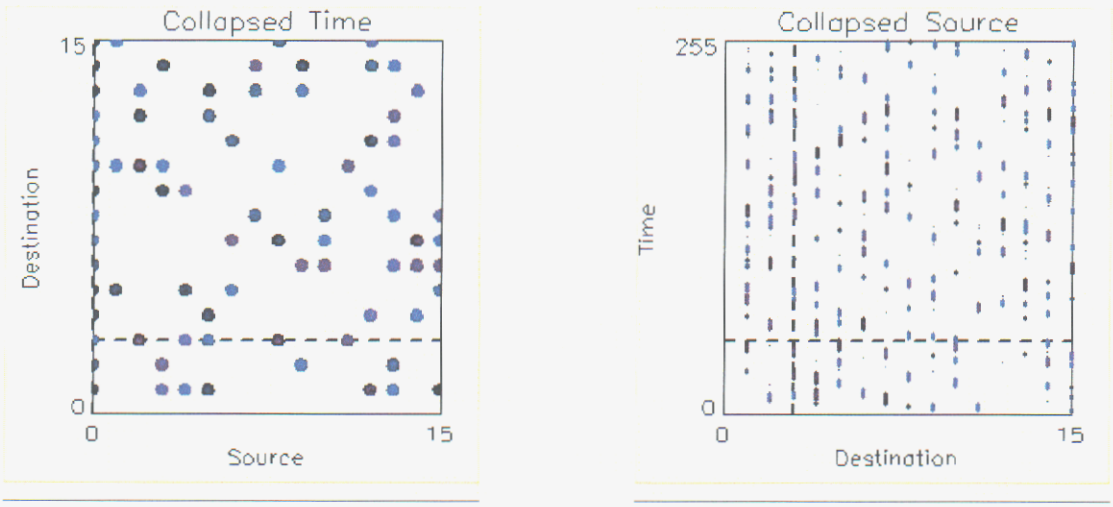

Figure 4. A collapsed time plot (left) and a collapsed source plot (right).

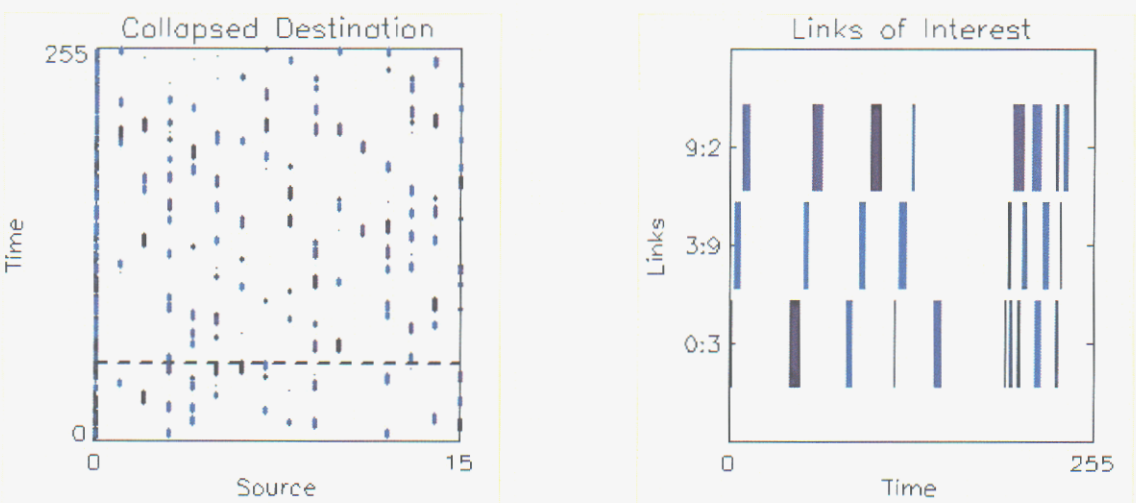

Figure 5. A collapsed destination plot (left) and a path or "links of interest" plot (right).

This visualization tool was successful in showing that there are many alternative ways of viewing the data. We showed that color could be successfully used to convey additional information to the user. However, the 2-D representation required that the user continually change views and did little to reduce the difficulty of discovering interesting multi-link relationships.

The 3-D Tool

Our second visualization tool was developed with Flatland, a 3-D visualization system developed by the Albuquerque High Performance Computing Center (AHPCC) at the University of New Mexico (UNM). Flatland runs on a variety of platforms and operating systems. It supports a full range of 3-D features, including hidden surfaces, volume rendering, and transparency, and allows the user to "fly through" the display. Flatland source is available for open distribution. We contracted with UNM to develop a Flatland application module to support this project. Flatland provides capabilities that are unavailable in IDL.

Methods for Visualizing Data

Method 1: The Raw Data

Figure 6 shows the most basic display of the data possible – a window with a single record. The data are made up of many such records, each describing a transaction between two nodes. Each transaction has a record number, the input (sending) node and output (receiving) node, the start time, and a duration.

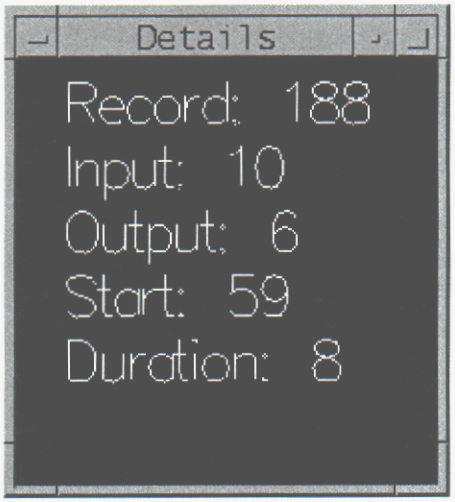

Figure 6. Details of a raw record.

Any record of the input data can be viewed this way. This method has the advantage of being very precise, but it is also quite slow and cumbersome and relies entirely on the user to form a mental or hand written picture of the data useful in the search for paths.

The raw data is, therefore, not conducive to finding paths. Even if none of the information were missed, the user would have to use a pencil and paper to keep track of possible paths. In current practice, most analysis is done with text-based displays. A query returns a list of possible entity transaction entity sets (also referred to as arcs or node sets). A query by nature filters the raw data. Subsequent queries can further filter raw data for one-transaction-at-a-time analysis. For instance, one can specify a node, call it Node A, requesting all possible paths. Suppose it had paths to "B," "C," and "D." The query can ask for other nodes that have paths going through "B," "C," and/or "D." The results will return a text-based display that shows new nodes and which nodes it has in common ("B," "C," and/or "D") with Node A.

B. Method 2: Circle Graph

In Figure 7, we see a graphical representation of the system of transactions. The window on the left shows the 16 nodes in our example data with all transactions. There are many transactions in the data that occur between the same nodes at different times, which are not

detectable in this diagram. The direction of each transaction also is not discernible from this representation.

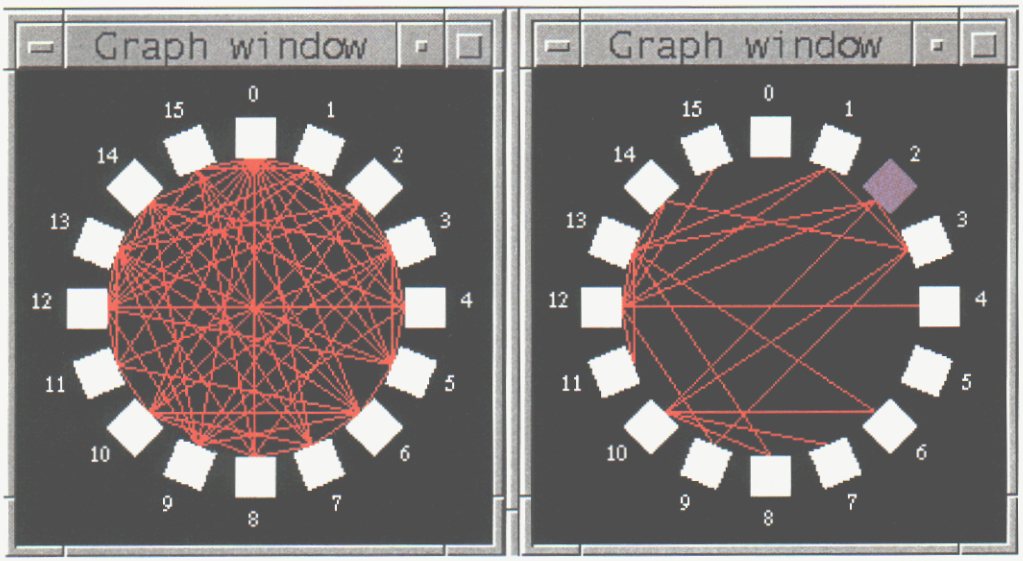

Figure 7. A circle graph representing transactions.

In Figure 7, the graph on the left represents all the data returned from a query. In this example all possible paths start at Node 0. These paths can be any length. The circle graph showing all connections illustrates nodes that do not have transactions between them, e.g., 6 and 12, or 9 and 10. The user may also determine which entities have no transactions between them if the graph does not have too many arc crossings.

This graph view also supports filtering. The user can select any node and then apply a filter that limits the display to transactions that are "N" links from the selected node, where the analyst specifies "N", and a link is a transaction from one entity to another. The "N" links must be temporally sequential, and the source node for the later transaction must be the same as the destination node for the previous transaction.

The right window in Figure7 demonstrates a filter that limits the display to transactions that start at Node 2 and are not more than two links from it. This view shows the activity between nodes and demonstrates which nodes have the largest number of transactions with other nodes.

With the circle graph display, the user can determine which nodes have paths between them and the length of the paths by successively looking at paths of various lengths. In both the filtered and unfiltered views, the only temporal information is that a successive link must have a start time after the completion of the succeeding link. However, temporally, these links could be years apart, and transaction patterns are hard to detect. The representations need to include more meaningful temporal data to find the patterns in which we are interested.

Method 3: Adjacency Matrices

An adjacency matrix is a good way to represent arbitrary transactions. The standard adjacency matrix would be a slice or volume of time with the input nodes on one axis and the output nodes on an orthogonal axis, a link of length 1. If all possible time slices of the data were stacked with later slices on top of earlier slices, the data would be contained in a volume with inputs, outputs, and time on orthogonal axes in three dimensions. These two slices show all transactions entering or exiting a given node, respectively. The data then can be just as easily sliced between time and the inputs (senders) or time and the outputs (receivers).

Figure 8 shows three different slices of data, with each slice parallel to one of the axes. The upper left window shows time on one axis and outputs on the other. The upper right once again shows time on the vertical axis, with inputs on the horizontal axis. The lower window of Figure 8 shows the traditional adjacency matrix. These slices can be moved through the data (the window title bars identify the slice location in these examples). To help with understanding, each slice is shown as a line on the other two representations. Color-coded rectangles represent connections. A spectrum from blue to red represents the duration value, which also is represented in the "Input Slice" and "Output Slice" windows by the height of the rectangle. Dotted lines show the intersection of the other planes relative to the view shown. These representations are similar to the ones shown in Figures 4 and 5 for the 2-D model.

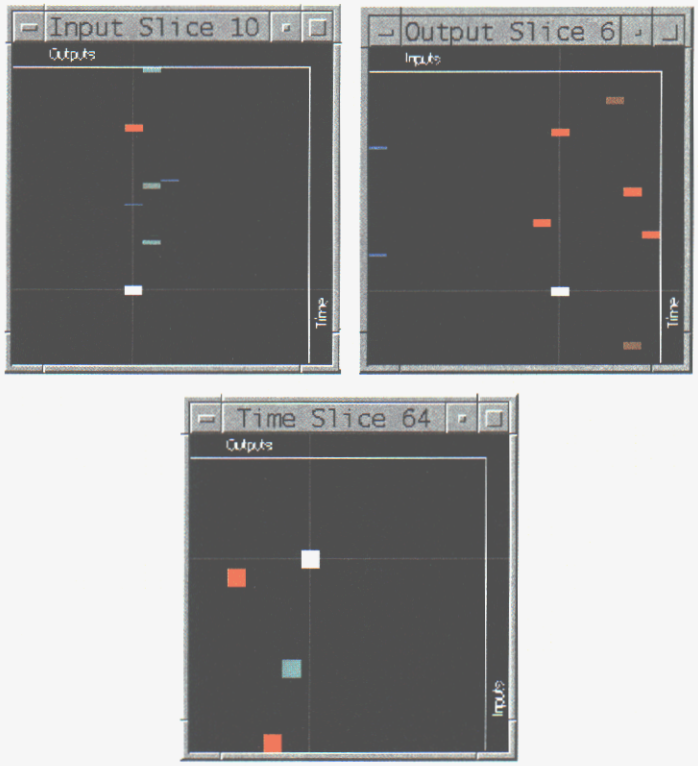

Figure 8. Three windows showing three orthogonal slices of the data.

The user has selected the highlighted (white) transaction at the intersection of all three slices. Data about the transaction is then displayed using Method 1, the raw data.

This method is an improvement over the previous two methods, as it allows the user to view the data along any of the three obvious axes. However, it still requires the user to do a tremendous amount of mental visualization and correlation. In order to find a path in this sort of representation, the three views can be used to show the relationships between the transactions, but it is still up to the analyst to relate different transactions and discover they belong to the same path. This relationship is not displayed explicitly.

Since these 2-D representations have limits, we looked at the 3-D volume associated with the 2-D slices.

D. Method 4: 3-D Adjacency Matrix

The mental model of Method 3 requires that the analyst keep track of time-related data independently of the visual representation. When we stack the tome slices with the input/output slices, the result is a 3-D adjacency matrix.

Figure 9 shows this stacked data mapped into a three-dimensional space, using the Flatland [7] virtual environment (VE). Flatland is a highly configurable research VE developed at UNM's AHPCC. Using shared libraries and run-time configuration, Flatland can be adapted to present a wide variety of environments and comes with standard example modules for the ground, stars, and a sun, and example modules to help developers construct their own Flatland module applications. Any visualization that uses OpenGL can be easily adapted to Flatland, and Flatland supports 3-D audio. Application menus and various interaction modules allow the user to interact with and manipulate objects in the environment. Using Flatland allows the user to fly around and into the data, as well as control the representation.

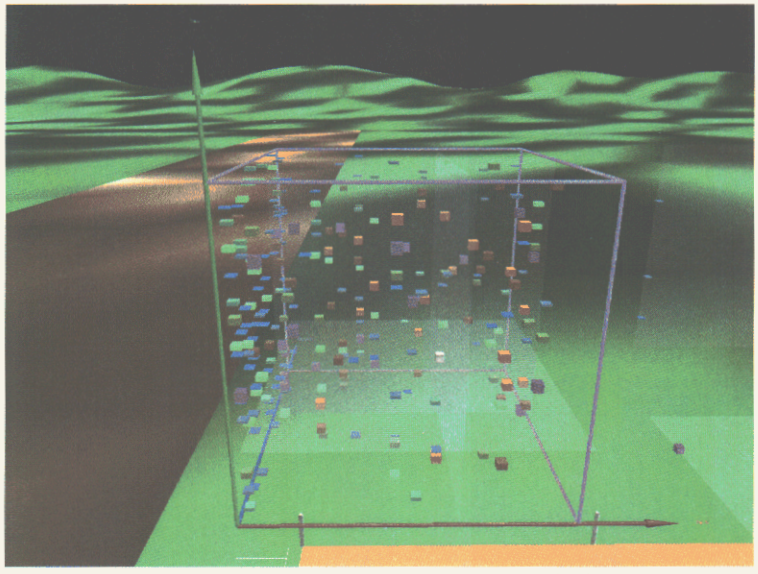

Figure 9. A 3-D mapping of the example data in the Flatland virtual environment.

In the Flatland representation of the data, inputs are mapped in the +X direction (left to right), outputs in the -Z direction (front to back), and time in the +Y direction (bottom to top). Slice planes (the translucent planes inside the box), similar to those used in Method 3, are shown for all three axes. Since it is difficult to see where the slice planes intersect transactions, especially when the user's view is not tangential to the plane, the slice planes are statically reproduced to one side. For example, the planes to the right of the volume are projections of the slice planes inside the volume, Figure 10. An interesting feature is that in the horizontal-slice plane (time), the transactions move through the plane, allowing a view of where in the duration of a transaction (the white box) the slice cuts each individual transaction. This is an added piece of information unavailable directly in the time-slice view in Method 3. Note also that the highlighted transaction described in Method 3 also is highlighted in both the cut planes and the 3-D adjacency-matrix views.

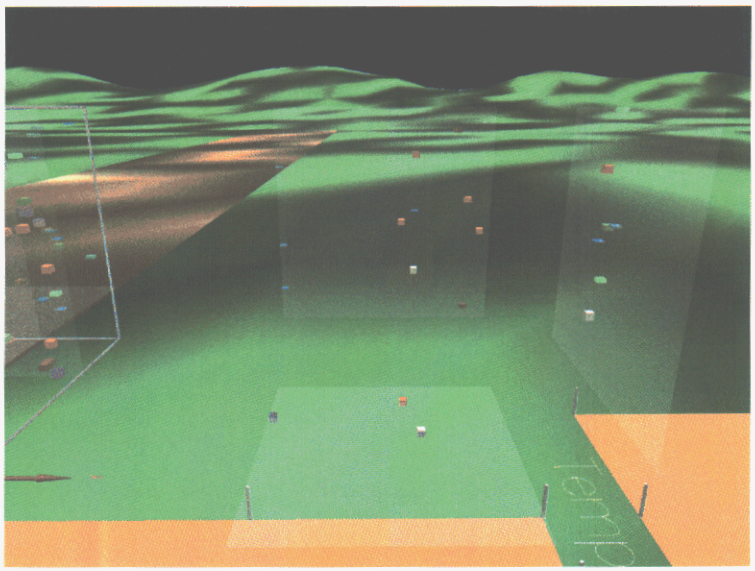

Figure 10. Static representations of the slice planes seen in Fig. 9.
Instead of moving through the data, the same data being sliced in the volume appears embedded in these nonmoving planes.

In this representation, the analyst can control the location of the slice planes in the data, the location of the static cut planes (left or right side, top or bottom, and front or back), and whether the slice and static cut planes are displayed at all. Additionally, the display of the axes and the transaction element size can be controlled interactively.

In this representation, the analyst can see all of the data at once, and even fly in and look "close up" at a specific section. Transaction patterns, especially in long chains of transactions, are still not clearly discernable partly because most of the data is noise for our analysis and partly because of the inherent limitations of the adjacency-matrix format. The adjacency matrix requires the user to trace a connection from an input to an output, find the output node back on the input axis, then repeat, Figure 11. This process is tedious and error prone, and if the paths branch frequently, the number of possible paths increases quickly, making tracing difficult.

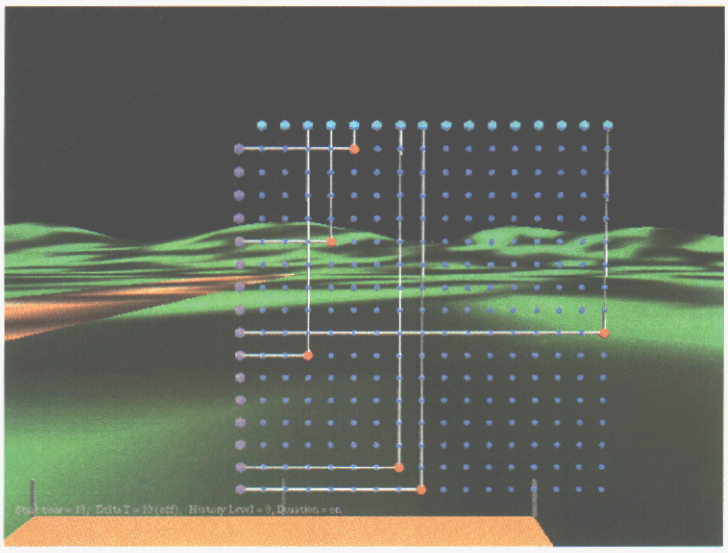

Figure 11. A standard adjacency matrix, rendered in 3-D.

The slice data, 3-D adjacency matrix, and the 3-D slices are not very effective for finding paths. First, the paths are obscured by a large amount of noise. In addition, the paths we are looking for cross time and input and output plane projections. Thus, even if the noise was suppressed, a 2-D projection does not contain enough information. For instance, to find all the paths from Node A to Node B, we need to span both time and the planes for all the nodes through which the path from "A" to "B" passes. A display showing time on one axis and arcs between nodes on a given path on another axis is more conducive to identifying patterns.

The filtering process described for Method 2 can also be applied to the 3-D adjacency matrix. This filtering reduces the data to a manageable number of transactions, but it is not selective enough to point out individual transaction chains. To overcome these difficulties, new representations need to be considered.

Method 5: Parallel Axes Graph

The parallel axes graph is specifically designed to find transaction paths in the data. It allows the analysis to specify a source (beginning) node, a destination (ending) node, and the number of transactions, or links, between them. For instance, the analyst could elect to see all paths from Node 3 to Node 5 that occur in exactly five transactions.

There are two primary modes for this display. The first includes minimal temporal information. This mode only shows the nodes in a path, not when the transactions occurred (even though successive transactions must be later in time than the previous transaction). The data displayed in Figure 12 actually corresponds to seven (possible) paths of interest. In the figure, only five distinct paths are recognizable since an identical path occurring later in time is overlaid on an earlier path. These paths need to be separated by start time (along the Z-axis,

or some other time parameter) to make them distinct. Figure 13 shows the second mode of display with the paths projected along a time axis. Each transaction is marked with its duration using a banner extending in the -Z direction from the beginning of the transaction. This banner is drawn in cooler (blue) colors for short durations and warmer colors for longer durations. The width of the banner also is indicative of the duration, with narrower banners for shorter durations and wider ones for longer durations.

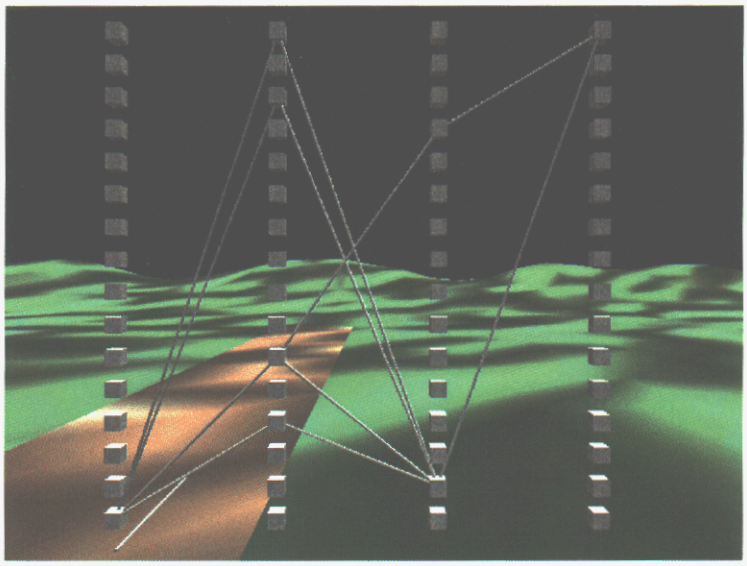

Figure 12. Parallel axes graph without temporal and duration information
A representation of all paths in the data from Node 0 (the bottommost node) to Node 15 (the topmost node) of Length 3, at a start time of 0, and a Delta "t" of 68.

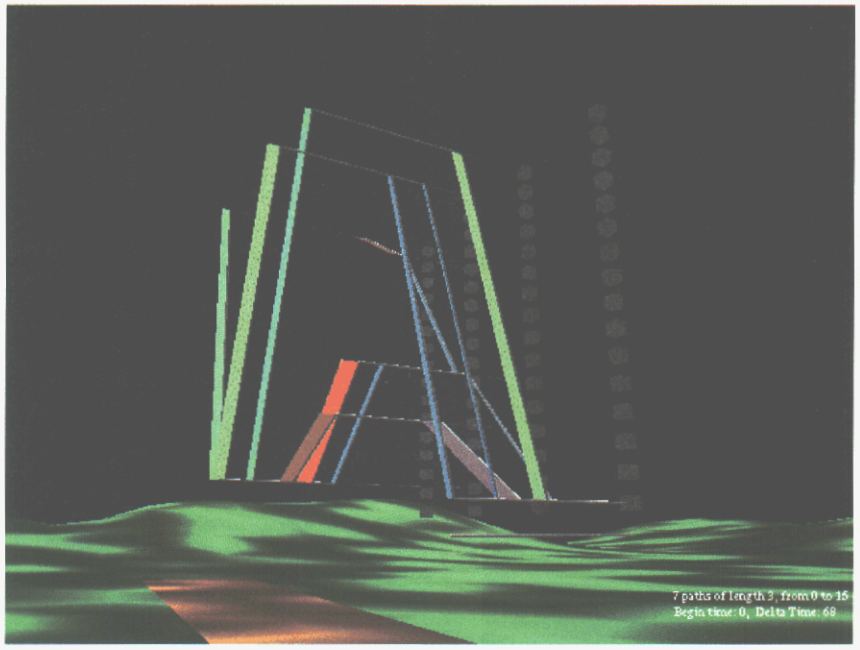

Figure 13. Parallel axes graph with temporal and duration information
The same data as displayed in Fig. 12, with temporal and duration info added.

A small line is drawn from each transaction terminus to the node placeholders at the front of the display. This makes it easier to line up the nodes with the corresponding transaction.

Two other parameters that can be used to filter the data in this view are start time and Delta "t". Start time is the earliest time value for which data will be displayed. The start time used in the creation of Figs. 12 and 13 was 0, our relative start time. This interactive filtering simplifies the visual representation.

Delta "t" is the allowable idle time at any given node. In other words, a transaction A is considered to be part of a path if it starts no longer than Delta "t" time units after another transaction, "B," if the output of transaction "A" is the same as the input of transaction "B." This reduces a display of hundreds or thousands of possible paths to fewer possible paths since it filters out paths that have too long a time lag before the next link

The bar just below the first column of nodes graphically represents the beginning time plus Delta "t." This visual aid helps keep track of the time interval being considered.

With this representation, if a path is displayed, the analyst can see there is a path from the input to the output node of a particular length. It also provides a start for finding particular entities and transactions that are of specific significance in completing a path. Such entities and transactions can be repeated as different paths are explored.

Paths are found by following transactions that start at the beginning node and recursively finding out where they lead. Paths have to start between the start time and the start time plus Delta "t." For a transaction to be considered as part of a path, it must occur within Delta "t" of the last transaction and exit from the last node entered. A path is only found if it terminates at the ending node and is the specified length.

There are two additional arrangements for displaying the nodes in this method. Neither configuration allows for the display of temporal data, thus limiting the usefulness of these displays in this application. One arrangement shows the nodes at each step arranged on a square grid. The transactions are drawn between these grid planes. The second arrangement places the nodes in a circle. These circles of nodes form a cylinder inside of which the transactions are drawn, Figure 14.

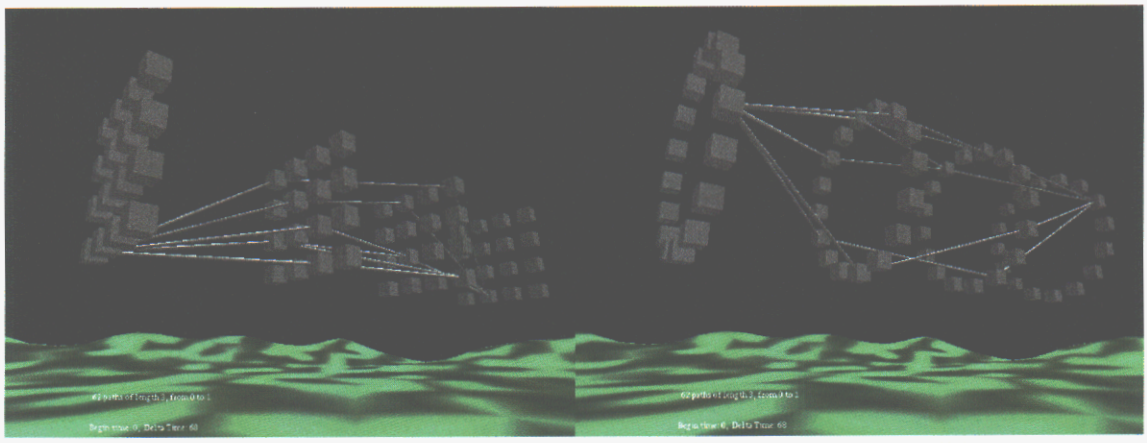

Figure 14. Two alternate representations of the parallel-axes graph.

The parallel axes graph seems to be the most intuitive representation for finding paths of a given length between a specific starting and ending point. The user can control the length of the path, the starting node, the ending node, the length of time displayed on the graph, and the time between successive links. This allows for a representation that has filtered out much of the noise so that smaller amounts of data are shown. One also can scroll across time looking for repeated patterns by successively increasing the start time.

Method 6: Dreamcatcher

The Dreamcatcher method derives its name from the Native American Dreamcatcher art. It is another representation of an adjacency matrix.

One of the primary difficulties with an adjacency matrix is tracing successive links from input to output. Finding a path requires one to follow a connection from an output to an input, find the output corresponding to the input, and then start again. See Figure 11 for an example of this process.

If we take the standard-adjacency matrix depicted in Figure 11 and rotate the input nodes axis (along the top) 270° counterclockwise around the origin (the upper left corner), leaving the output nodes fixed, then rotating the intermediate points along curves of constant radius and linearly interpolating their individual angles between those two axes, we get the construct in Figure 15. Possible paths are now quite visible as continuous connected loops, and the graph now resembles its namesake.

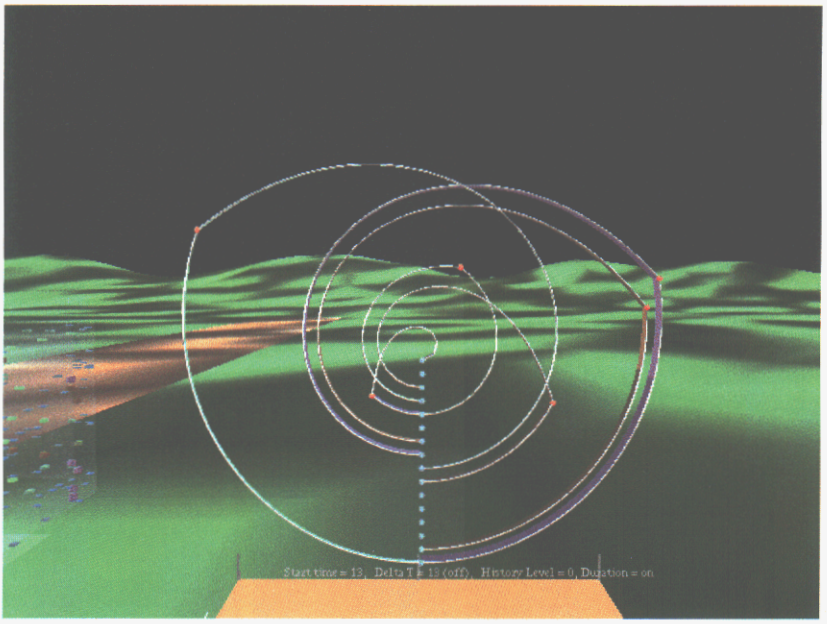

Figure 15. The standard-adjacency matrix (Fig. 11) with rotated input notes
The input nodes are rotated 270° counterclockwise. Now the input and output nodes are coincident and possible paths are more easily seen.

In order to facilitate the search for paths in the data, the Dreamcatcher has several features that are all capable of being set dynamically.

The unused intersection points (those not at a transaction intersection) can be removed, as they can be distracting.

The Dreamcatcher can be folded and unfolded at any time. The fold/unfold sequence is animated, so the user can see how the unfolded representation is derived from the adjacency matrix.

The "grid lines" (the pipes representing the transactions themselves) can be removed, but this representation only leaves the sphere at the transaction intersections, which is not particularly useful in this application. These connection joints remain when the unused intersection points are turned off, and they can also be a different color than normal unused intersection points.

The Dreamcatcher provides a history function. Rather than have a transaction abruptly disappear when it is no longer active in the time slice, the transaction fades, using transparency, according to how far in the past it was active. For instance, if the current time is 32, and a transaction ended at time 30, it would fade an amount corresponding to two time units. The fade duration is controllable and can be set from zero (no history function) to the maximum time represented in the data.

The Dreamcatcher, like the method in the previous section, can represent the duration of the transaction using banners trailing from the grid lines. These banners are width and color-coded according to the duration of the transaction – narrower, cooler colors for short durations and wider, hotter colors for longer durations. These widths and color spectrum are normalized over the range of the data set being displayed.

Another dynamic setting is the pulse generator that generated pulses traveling the grid lines counterclockwise. The idea was to encourage the eye to follow the lines as they looped around the structure. The designers all agree that this is very distracting and visually busy. The feature still exists, but defaults to "OFF."

Transactions can be chosen for display in one of two ways. Either the transaction is displayed just when the selected time value is the same as the transaction start value, or the transaction can be displayed as long as the selected time falls within the duration of that transaction. For finding paths, the latter mode is more useful, since having transactions visible after their start time often leads to false identification of paths that do not exist.

The concept of Delta "t," as described in the previous method, also is used to filter data in the Dreamcatcher. When enabled, all transaction paths that have no more than Delta "t" time units between incoming and outgoing transactions at every node in the path are displayed. Whenever a path of two or more transactions are found, the entire path turns orange to differentiate it from chance loops that do not qualify as paths. This is illustrated in Figure 16.

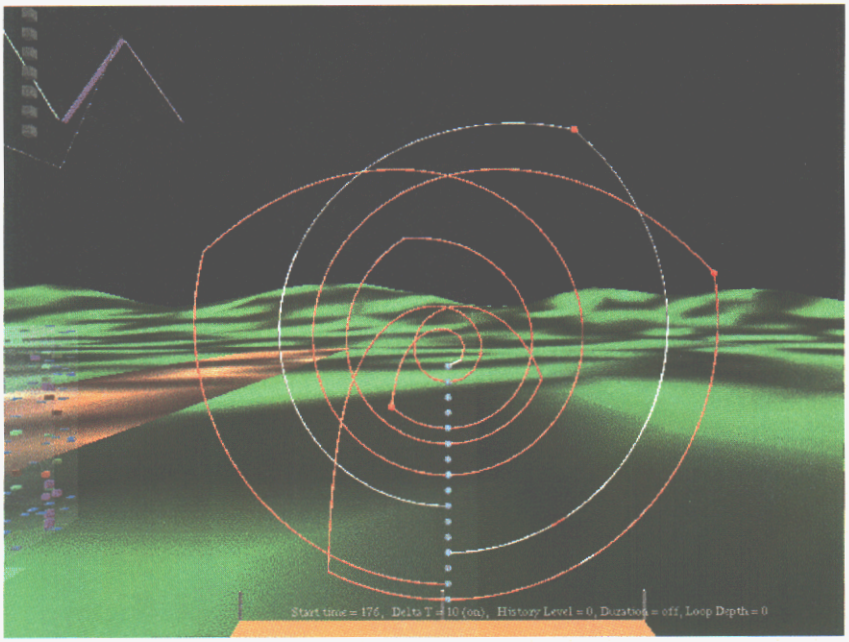

Figure 16. Delta "t" enabled.
The paths found are emphasized with a darker color, while transactions not participating in a path are white.

Occasionally, apparent paths can form even when none are present. This happens when a path ends on the same node on which another path starts (in this situation the starting path starts at the currently selected time, while the ending path is arriving at that node some time in the future—they are not temporally coincident). Figure 16 illustrates this, also. The apparent path that goes through Node 15 (the bottommost node) is not, in fact, a path. It is the termination of one path and the start of another.

To help resolve this issue we have introduced "rabbits," named after the electric lights used on airport runways to guide aircraft in for landing, which originated after the mechanical rabbit used at dog racing tracks. These rabbits are launched at the start of every transaction that is not a continuation of a path (i.e., the start of a path or a single transaction) and circle the Dreamcatcher counterclockwise along the transaction pipes at a constant angular rate. The current depth of the rabbit(s) (the "depth" of a rabbit is how many times the rabbit has touched a node, and is, therefore, the path length it has covered so far) is displayed at the end of the textual information at the bottom of the figure and relates the length of the path in hops from the start.

The rabbits are represented by a bead on the connection pipe that is slightly larger than the pipe itself and has a complementary color. Figure 16 shows an example containing rabbits. This image was taken immediately after the rabbits were launched. The white and the orange transactions, obviously the first transaction in a path (or, in the case of the white one, a single transaction), have a rabbit running. The transaction from Node 15 (the bottommost node),

which appears to be in the middle of a path, also has a rabbit running. This indicates that there is a path starting in what appears to be the middle of a path. There must be, therefore, a path starting at time 176 at Node 15, a conclusion that would not have been immediately obvious by looking at the graph alone.

The "dumbbell" mode is also provided to help detect false paths. In this mode, the input and output nodes are slightly separated, and a bar is drawn across the resulting gap when a path passes through that node. Figure 17 shows an example where this is useful.

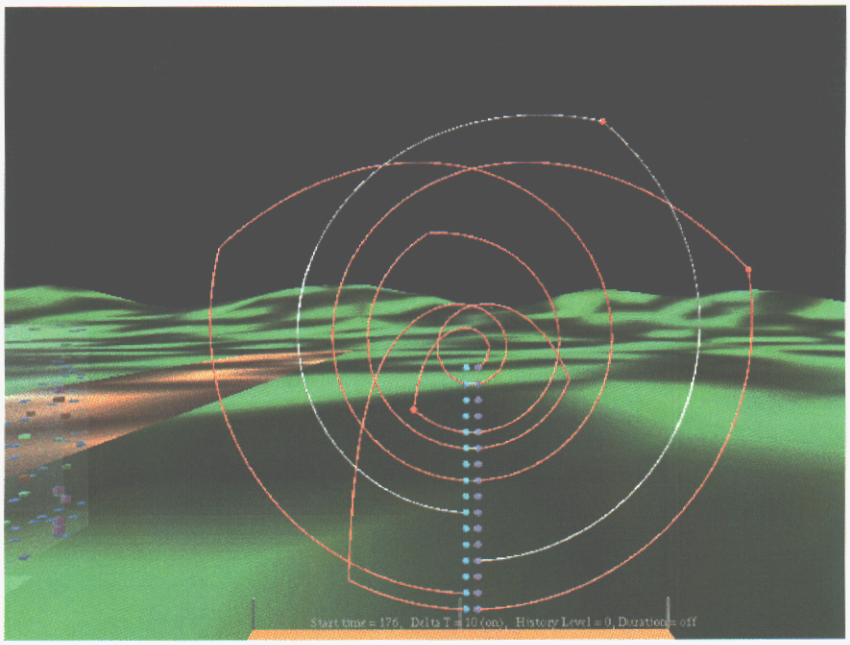

Figure 17. Delta "t" and dumbbell mode enabled
Showing the same data as Fig. 16. Notice that what appeared to be a path through Node 15 (the bottom-most node) is in fact the end of one path and the beginning of another.

In the adjacency-matrix view of the Dreamcatcher, it is difficult to trace paths, although it confirms the existence of paths. The unfolding facilitates the identification of paths. It appears to be useful for identifying possible paths of interest. It is less intuitive to determine if paths are repeated. The ability to cycle through time may be useful for qualitative measures of a particular node's activity.

This Page Intentionally
Left Blank

Conclusions

Analysts of this type of data currently employ a text-based analysis method using only the raw results of the database queries. They have begun some exploration into using 2-D mappings of the data without any temporal component, but are still in the experimental stage. Often analysts are not cognizant of the fact that they are looking for patterns. The research presented here is more advanced than the analyst's current tools, so analysts will require a great deal of training before they will be able to use these tools to their fullest potential. We anticipate that people who are unfamiliar with graph theory in interpreting the displays will encounter some difficulties.

Our 3-D tool is very robust, but its user interface needs additional work. The navigation controls are not intuitive, and it is easy to get hopelessly lost. Navigation by joystick is much more intuitive, but currently this capability is only available under Microsoft Windows. We would like to incorporate a heads-up display (HUD) to provide the user with orientation information. We would like to support both a control-key interface (such as it has now) and a menu-driven interface (for use by novice users).

We would also like to incorporate the "links of interest" display from the 2-D tool into the 3-D tool. This display made it easy to pick out recurrent patterns in the 2-D tool. We believe that it would have utility in the three-dimensional tool, too.

Finally, although the tools seem to make it easier to detect interesting connection paths, we would like to go a step further by automating the detection of interesting, multi-link paths. Initial work looks promising, but it is still far from complete.

This Page Intentionally
Left Blank

References

[1] S.E. Flynn, Beyond Border Control, in *Foreign Affairs*, vol. 79(6), pp. 57–68, 2000.

[2] L. Tweedie, Characterizing Interactive Externalizations, in *Proceedings of the 1997 Conference on Human Factors in Computing Systems*, Computer Human Interactions, Atlanta, Ga., USA, March 22–27, 1997, pp. 375–382.

[3] S.K. Card, J.D. Mackinlay, and B. Shneiderman, *Readings in Information Visualization—Using Vision to Think*, Morgan Kaufmann, 1999, pp. 308-309, 523, 525-526, 619.

[4] G.G. Robertson, S.K. Card, and J.D. Mackinlay, Information Visualization Using 3D Interactive Animation, in *Communications of the ACM*, vol. 36(4), pp. 57–71, 1993.

[5] K.L. Summers, J. Greenfield, and B.T. Smith, A Survey of Parallel Program Performance Evaluation Techniques Using Visualization and Virtual Reality, in *Proceedings of the IEEE Aerospace Conference*, Big Sky, Montana, 2000.

[6] A. Inselberg, Multidimensional Detective, in *Proceedings of InfoVis '97*, IEEE Symposium on Information Visualization, IEEE Information Visualization, 1997, pp. 100–107.

[7] A Guide to Flatland, unpublished Albuquerque High Performance Computing Center Technical Document, 2000.

Distribution:

1 Dr. Kate Cherry
Lockheed Martin
MP-166
12506 Lake Underhill, Orlando FL 32825

3 University of New Mexico
Attn: Kenneth L. Summers
Timothy Eyring
Thomas P. Caudell
Albuquerque, NM 87131

1	MS 1202	Roxanna M. Jausma, 5931
1	1217	William M. Miller, 5913
1	1205	K. David Nokes, 5900
1	1207	J. Yoder, 5914
5	1207	M.W. Trahan, 5914
10	1207	E.A. Walther, 5914
1	9018	Central Technical Files, 8945-1
2	0899	Technical Library, 9616
2	0619	Review and Approval Desk, 9612, For DOE/OSTI