Network-based Visual Analysis of Tabular Data

Zhicheng Liu* Shamkant B. Navathe† John T. Stasko‡

Georgia Institute of Technology

ABSTRACT

Tabular data are pervasive. Although tables often describe multivariate data without explicit network semantics, it may be advantageous to explore the data modeled as a graph or network for analysis. Even when a given table design conveys some static network semantics, analysts may want to look at multiple networks from different perspectives, at different levels of abstraction, and with different edge semantics. We present a system called Ploceus that offers a general approach for performing multi-dimensional and multi-level network-based visual analysis on multivariate tabular data. Powered by an underlying relational algebraic framework, Ploceus supports flexible construction and transformation of networks through a direct manipulation interface, and integrates dynamic network manipulation with visual exploration for a seamless analytic experience.

Index Terms: H.5.2 [Information Systems]: Information Interfaces and Presentation—User Interfaces;

1 INTRODUCTION

Network visualizations, often in the form of node-link diagrams, are an effective means to understand patterns of interaction between entities, to discover entities with interesting roles, and to identify inherent groups or clusters of entities. Many existing approaches to network visualization and analysis assume a given graph and static network semantics. During an analysis process, however, selecting, filtering, clustering or computing metrics over a static network is not always enough. Analysts may want to construct new networks and transform existing ones to explore the data from different perspectives and at different levels of abstraction.

The goal of our research is to provide a general approach for performing multi-dimensional and multi-level network-based visual analysis. We choose tabular data as the input data model considering the dominance of spreadsheets and relational databases in current data management practices. As we discuss below, tabular data may or may not contain explicit network semantics, and its multivariate nature implies the need of dynamic network modeling for greater analytic power.

1.1 Forms of Tabular Data

Tabular data come in many forms, each unique in its schematic and semantic structure depending on the technology used and the data owner's goal. The term "tabular data" is thus fairly broad and can be interpreted as either *multivariate data* or *attribute relationship graphs*. We give examples of different types of tabular data in this section, and will base our discussion on these examples throughout the rest of the paper.

Single tables are pervasive in the form of spreadsheets and comma-separated value (csv) files. For example, Table 1 shows visits to the White House. For each visit, it records the last and first name of the person arranging the visit (LName, FName), the type of visit (Type), the date (Date) and location (Loc) of visit, the size of the visiting group (Size), and the visitee's name (Visitee). Such tabular data are essentially *multivariate data* where rows represent entities or facts and columns represent entity attributes or other entities. In multivariate data, explicit network semantics are typically absent.

ID	LName	FName	Type	Date	Loc	Size	Visitee
1	Dodd	Chris	VA	6/25/09	WH	2018	POTUS
2	Smith	John	VA	6/26/09	WH	237	Office Visitors
3	Smith	John	AL	6/26/09	OEOB	144	Amanda Kepko
4	Hirani	Amyn	VA	6/30/09	WH	184	Office Visitors
5	Keehan	Carol	VA	6/30/09	WH	8	Kristin Sheehy
6	Keehan	Carol	VA	7/8/09	OEOB	26	Daniella Leger

Table 1: A table of sample visitor information to the White House

Multiple linked tables are pervasive in the form of relational databases, although the same tables can also be described in spreadsheets. In a relational database, the ER Model (Entity-Relationship Model [16]) typically underlies database design. Each row in a table represents a fact that corresponds to a real-world entity or relationship. For example, Table 2(a) represents facts about employees in a company, and Table 2(b) represents facts about departments in the same company. The two tables are linked by a *one-to-many* DEPARTMENT – EMPLOYEE relationship type. That is, one department can have multiple employees, but one employee can only work for one department. *One-to-many* relationships are typically captured by foreign keys in a relational database [18]. In this case, Dpt in the EMPLOYEE table is a foreign key, referencing the DEPARTMENT table.

(a) EMPLOYEE

ID	FName	LName	Bdate	Dpt
1	John	Smith	1965-01-10	2
2	Franklin	Wong	1952-04-09	3
3	Jennifer	Wallace	1970-10-23	3
4	Ahmad	Jabbar	1945-11-02	1

(b) DEPARTMENT

ID	Name	City	State	Latitude	Longitude
1	Headquarters	Los Angeles	CA	34.05	-118.24
2	Administration	San Jose	CA	37.34	-121.89
3	Research	Houston	TX	29.76	-95.36

Table 2: Two tables describing employees and the departments they work for

Another type of relationship in the ER model is the *many-to-many* relationship, and it is captured by a separate relationship table [18]. For example, Table 3(a) represents selected facts about research grants awarded by the National Science Foundation (NSF) in the Information & Intelligent Systems division, and Table 3(b) represents facts about researchers. The two tables are linked by Table 3(c), which represents a many-to-many "work-on" relationship. That is, one researcher can receive multiple grants, and one grant can involve multiple researchers too.

*e-mail: zcliu@cc.gatech.edu

†e-mail: sham@cc.gatech.edu

‡e-mail: stasko@cc.gatech.edu

(a) GRANT

GID	Title	Program	Program Manager	Amount
1	Data Mining of Digital Behavior	Statistics	Sylvia Spengler	2241750
2	Real-time Capture, Management and Reconstruction of Spatio-Temporal Events	Information Technology Research	Maria Zemankova	430000
3	Statistical Data Mining of Time-Dependent Data with Applications in Geoscience and Biology	ITR for National Priorities	Sylvia Spengler	566644

(b) PERSON

PID	Name	Org
1	Padhraic Smyth	University of California Irvine
2	Sharad Mehrotra	University of California Irvine

(c) WorkOn

Person	Grant	Role
1	1	PI
2	1	CoPI
2	2	PI
1	3	PI

Table 3: Tables describing researchers and the grants they receive

These tabular data in multiple linked tables are essentially *attribute relationship graphs* with explicit network semantics. Table 2 describes connections between employee and department entities. Similarly, Table 3 is a graph specifying the connection between two types of entities, researcher and grant, each with its own attributes.

An OLAP (Online Analytical Processing) database, unlike spreadsheets and relational databases, is not built for low-level atomic operations such as insertion and update but for analytical purposes. It uses data cubes for better performance in operations such as slice/dice and roll-up/drill-down. The analytical power of OLAP, however, is not necessarily suitable for network-based analysis because it focuses only on inherent relationships between entity attributes, and assumes different entities are mutually independent [15]. As a result, the OLAP framework is not directly relevant for our purpose, and in this paper we focus on spreadsheets and databases which provide a basis for an alternative network-centric framework.

1.2 Analytical Gap and Semantic Distance

For visualization designers and analysts, spreadsheets and databases naturally become the infrastructure upon which higher level visual analysis is accomplished. As discussed in the previous section, multivariate data in the form of single tables do not contain explicit network semantics; even when multiple tables are used to describe a graph, analysts' own notions of a meaningful network may render different graph structures. First of all, the concept of an entity is often multi-level nested. An attribute of an entity may be treated as an entity in its own right. For example, in Table 3(a), each row represents a grant entity with its own attributes such as title and program manager. A program manager can be in turn treated as an entity. In fact, it is often difficult to determine whether something is an entity or an attribute in data schema design [13]. Secondly, the same two entities can be connected via different semantics. In Table 1 for example, two people can be connected if they visited the same location, have the same last name, or started their visits on the same day.

The multivariate nature of tabular datasets thus implies opportunities for asking interesting questions that can be answered with network visualizations, and it is worthwhile to examine the nature of such questions more closely. Given the dataset in Table 3 for example, a grant applicant may want to understand the hidden dynamics, if any, in the process of awarding grants to choose an appropriate application strategy. NSF officials will want to understand the impact of the IIS program on the awardee social networks and on the creation and diffusion of intellectual property to evaluate funding policy. Many questions can thus be asked, for instance:

- **[Q1]** *Is there a strong affiliation between program managers and research institutions? i.e. Do certain program managers tend to give awards to a few selected institutions only?*
- **[Q2]** *From which organizations do researchers tend to have more cross-institution collaborations?*

One possible way to answer Q1 is to construct a network visualization (Figure 1(a)) where an organization and a program manager are linked if the manager has awarded at least one grant to researchers in that organization. We can define the edge weight, to be the total grant amount as shown in the figure, or to be the number of grants awarded. Analysts can provide initial answers to Q1 by inspecting the overall connectivity of the network. If the network consists of multiple small subnetworks that are disconnected from each other, there is evidence that a strong affiliation does exist. It is also likely that there is no disconnection within the network, but certain organizations or managers occupy more central roles. Statistical measures will enhance visual inspection to provide a more precise assessment.

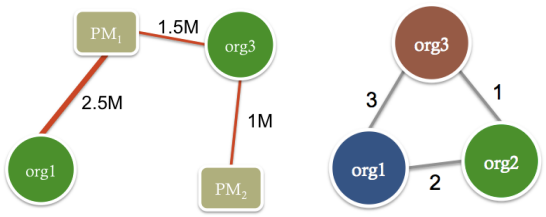


Figure 1: Visual models for answering questions on the NSF data set

Similarly, to answer Q2, we can create a network visualization where two organizations are connected by an edge if there is at least one collaboration between any researchers from these two organizations. Figure 1(b) shows this network semantics, where the edge weight is based on the frequency of collaboration. Applying an appropriate layout algorithm to this network visualization and using statistical measures such as betweenness centrality will likely reveal important organizations that are "gatekeepers" connecting different subgraphs.

These questions are examples of *high-level* analysis tasks [11]. These high-level tasks have two major characteristics. First, they cannot be answered satisfactorily by simple "yes" or "no" or some precise values and metrics. Analysts can define measures to quantify "affiliation strength", for example in the case of Q1, but such numbers are only meaningful at the level of specific manager-institution pairs. Network visualizations are useful to show global structures in the network. Secondly, these high-level tasks are semantically rich and context dependent, and cannot be described abstractly or captured *a priori* because they usually only emerge during the process of exploration.

These high level tasks can be compared with *low-level* tasks [25, 30], which are usually topology-based or attribute-based. Topology-based tasks include finding neighbors, counting degree, finding shortest paths and identifying clusters; attribute-based tasks include finding nodes with specific attributes, or finding nodes connected by particular type of edges. Many of these low-level tasks are well defined questions with clear-cut answers, and they can often be effectively answered using search or database queries without much visual representation.

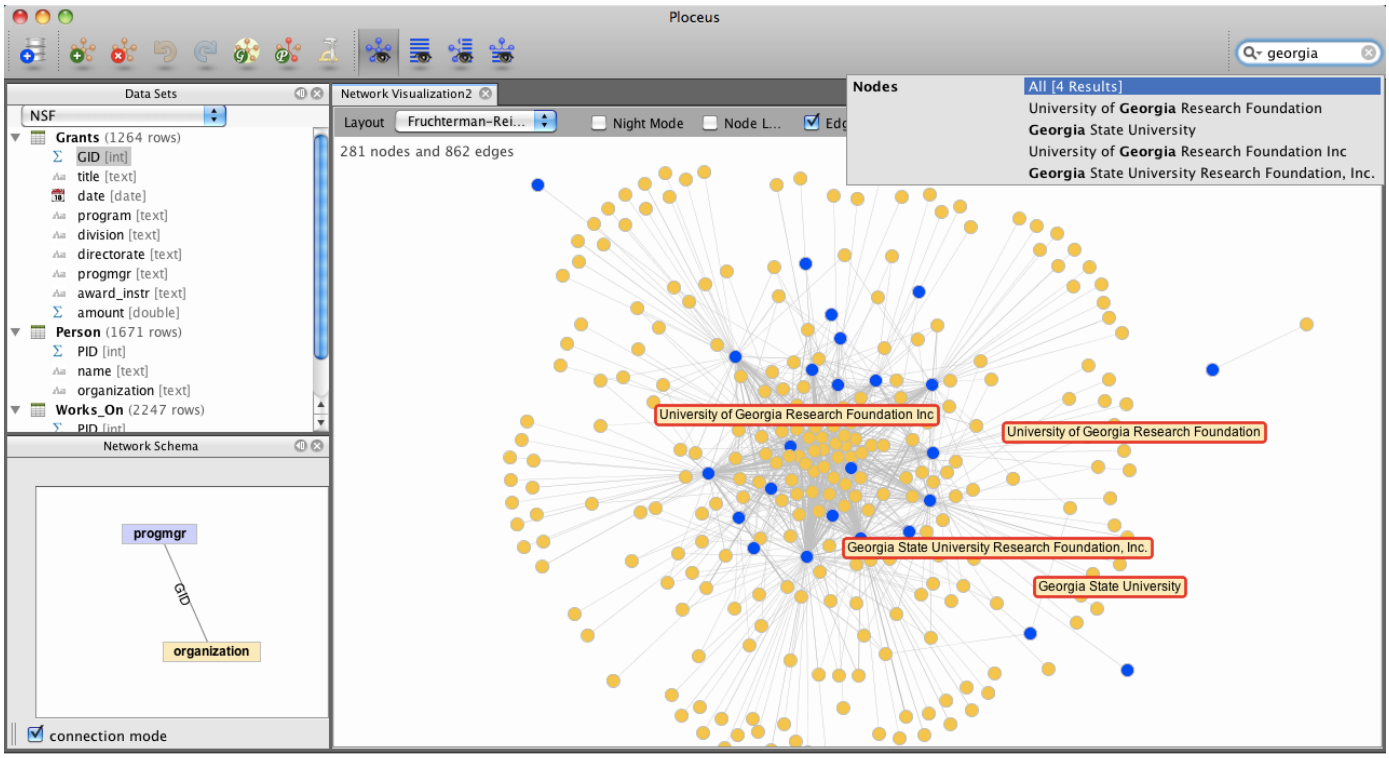


Figure 2: The Ploceus system interface with a data management view on the top left, a network schema view on the bottom left, and a network visualization view on the right.

Supporting only low-level tasks creates analytic gaps in addressing real analytic and sense-making goals. Many high level tasks require analysts to go beyond manipulating a static network and to actively construct and simulate a model [26]. Figure 1(a) and 1(b) are illustrations of analysts' desired model based on their analytical questions. To effectively support model-based reasoning, analysts must be able to quickly choose the relevant entities and relationships for model construction. The model will be subject to constant refinement and revision, where new variables and relationships are introduced and old ones transformed or discarded. Dynamic articulation of fluid network semantics is thus necessary, and the multivariate nature of many tabular datasets provides a fertile playground for performing this kind of model-based reasoning.

With these considerations in mind, we present Ploceus[1], a system designed to support flexible network-based visual analysis of tabular data. Our focus is not on representation and interaction techniques for visually analyzing a given network - a number of commercial and research systems have been designed for this purpose [2, 3, 5, 10, 21, 23, 28]. Rather, we aim to address flexible and rapid construction and manipulation of networks from tabular data. The power of Ploceus is based upon a formal framework that systematically specifies operators for network construction and transformation and the implementation of these operators in relational algebra. A direct manipulation interface is coupled with the formalism to help analysts articulate the desired network semantics.

2 Ploceus: Overview

Ploceus provides a direct manipulation interface for fast construction and transformation of networks, and shows immediate visual feedback on the network being created. Model construction and visual exploration hence are interweaved in a seamless manner. Ploceus contains three major views: a data management view on the top left, a network schema view on the bottom left, and a network view on the right (Figure 2). The data management view shows information about the columns in each table in a dataset; the network schema view is a sandbox-like environment where users can construct and manipulate networks at a conceptual level; the network view shows the corresponding network visualization and updates whenever the network schema is modified.

2.1 Operations

Ploceus currently supports the following types of operations. We describe these operations at a functional level in this section, and discuss the precise mechanisms of accomplishing these operations in Section 4.

- **Create Nodes**: Transform the values in one or more columns into node labels. For example, we can construct a set of nodes representing the people visiting the White House from all the rows in Table 1, and the labels of the nodes are created from the LName and FName columns. This results in four nodes: "Dodd, Chris", "Smith, John", "Hirani, Amyn", and "Keehan, Carol".
- **Add Attributes**: Transform the values in one or more columns as attributes of existing nodes. For example, we can add an attribute AccessType to the people nodes constructed from LName, FName earlier. The node "Dodd, Chris" will have the value "VA" for the AccessType attribute. Ploceus supports adding columns as attributes from a different table too. For example, we can add Role from Table 3(c) as an attribute for the Name nodes constructed from Table 3(b). Ploceus only allows a node to have one value for any particular attribute, so there will be two "Sharad Mehrotra" nodes in this case, one having a PI role and the other having a CoPI role.
- **Create Connections**: Create edges between existing nodes. For example, we can connect LName, FName nodes and Loc

[1] Ploceus is a kind of weaver bird that can build sophisticated nests.

nodes from Table 2 to see the visiting patterns by the visitors to the various locations. We can also connect nodes created from different tables, e.g. ProgramManager nodes from Table 3(a) and Org nodes from Table 3(b). When multiple tables are involved, Ploceus determines how the tables should be joined by analyzing the foreign key constraints between the tables through the Dijkstra shortest-path algorithm. In this case, the two tables are joined through Table 3(c). Ploceus computes whether there should be an edge between any two nodes as well as assigns a weight to that edge. When multiple ways of joining tables are possible, users can specify the join condition through a dialog.

- **Assign Weights**: Assign numerical weights to edges. Ploceus by default assigns a weight to each edge created, indicating the frequency of co-occurrence between the nodes in the data (Section 2.3 and 4 discuss edge weights in greater depth). For example, if we connect LName, FName nodes and Loc nodes from Table 1, by default the edge between "Dodd, Chris" and WH has a weight of 1, indicating this person has visited the White House once in this dataset. We may instead want to represent the connection strength by the number of people he has brought on his visits, and assign the column Size as the edge weight. The edge between "Dodd, Chris" and WH will have a weight of 2018. Only a single column can be assigned as edge weight, and that column must be quantitative.
- **Project**: Connect two nodes if they both are connected to the same node of a different type. Projection is a commonly used technique to reduce modalities of a network for analysis [24]. In a two-mode (i.e. there are two types of nodes) LName, FName - Loc network, for example, if "Dodd, Chris" is connected to "WH" (i.e. Chris Dodd visited the White House), and if "Keehan, Carol" is connected to "WH" also, after projecting LName, FName nodes on Loc nodes, "Dodd, Chris" and "Keehan, Carol" are connected. Figure 3(a) shows this process. The weight of edges after projection reflects the unique number of Loc nodes being projected.
- **Aggregate**: Group multiple nodes and treat them as one node. Ploceus automatically aggregates nodes with identical labels if no attributes are specified for these nodes, and aggregates nodes with identical labels and values if attributes are specified for the nodes. As a result, we have four distinct LName, FName nodes from Table 1, while there are actually six rows in the table.

  Other types of aggregation include but are not limited to the following.

  - *Pivoting*: PivotGraph [35] terms this operation *roll-up*. Given LName, FName nodes with the attribute AccessType, we can aggregate people nodes when they share the same AccessType. The pivoting process is visualized in Figure 3(b). The resulting graph shows the locations that are typically visited for different types of visits.
  - *Binning*: for nodes whose labels or attributes are derived from quantitative columns, value based aggregation is possible. One type of value based aggregation is *binning*: we divide the range from the minimum to the maximum attribute values into bins. For example, we can categorize Amount nodes created from Table 3(a) into three bins: "small" if Amount <= 500k, "medium" if 500k < Amount <= 1200k, and "large" if Amount > 1200k.
  - *Proximity grouping*: group nodes in a pair-wise manner if they have values close to each other. For example, from Table 2(b) we can create City nodes with attributes Latitude and Longitude. We can then aggregate every pair of City nodes into one for which the distance between them, computed from the latitude and longitude information, is within 500 miles. This operation is combinatorial: if there are four cities, and every one is within 500 miles to each of the other three, proximity grouping will produce $\sum_{k=1}^{(4-1)} k = 6$ nodes. Proximity grouping is useful when combined with projection, so that we can, for example, create a network of employees whose workplaces are within 500 miles to each other (to do this, connect employee names with cities, aggregate cities, then project employees on cities).

- **Slice 'n Dice**: Divide a network into sub-networks based on selected columns. For example, given that we have constructed a LName, FName - Visitee network from Table 1, we may want to see how the visiting pattern is related to the locations of visits by dividing the network using Loc slices. We will then have two subnetworks, one representing the visiting patterns at the White House ("WH"), and the other at the Old Executive Office Building ("OEOB"). Slice 'n dice thus enables analysts to create and organize meaningful snapshots of a big network based on different perspectives. The values in columns used for slicing and dicing are either categorical or can be categorized. When hierarchical categories exist, analysts can slice and dice at multiple granularities, e.g. for a *date* column: day → week → month → quarter → year.

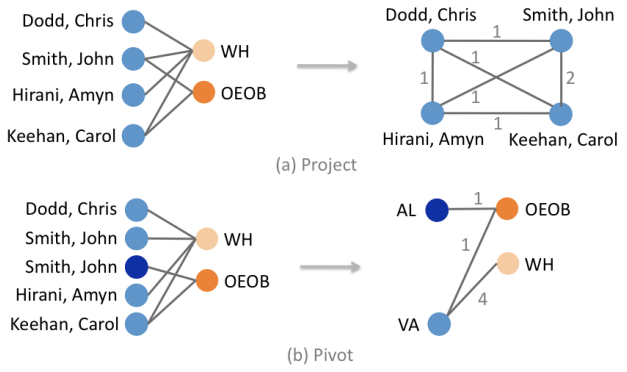


Figure 3: Project and Pivot Operations. In (a) the visitor nodes do not have attributes, and in (b) the visitor nodes have attribute "Type"

In addition to these higher-level operations for creating and transforming networks from data tables, Ploceus supports interaction with individual nodes such as selecting, filtering, moving, hiding, showing and expanding (showing neighbors of a node), interaction with the visualization in the form of zooming, panning, adding new visualizations and deleting existing visualizations, applying various network layout algorithms and analytical measures such as node degree, shortest path, betweenness centrality and closeness centrality. These features, though not the main focus of our research, are essential for integration with the above operations for seamless data transformation and visual exploration.

2.2 Design of Direct Manipulation Interface

The major design challenge in building Ploceus is *how to reduce articulatory distance*, i.e. assuming the analysts want to perform some operations, what is an intuitive way for them to communicate the intent to the system.

We chose the direct manipulation paradigm as our main design approach. To create nodes, analysts drag and drop selected columns from the data management view to an empty area in the network

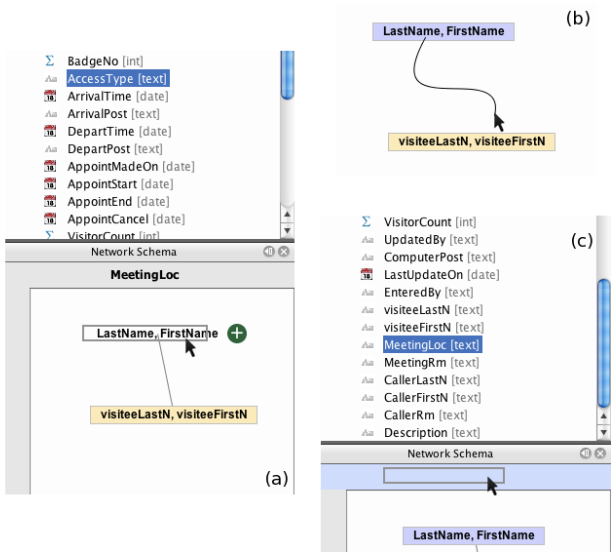


Figure 4: Direct manipulation interfaces for various operations: a) add attributes, b) create connections, and c) slice 'n dice.

schema view. Each drag-and-drop action creates a type of node, and the system assigns a color to that type. Dragging and dropping columns on top of an existing node type add those columns as an attribute to the node type (Figure 4(a)).

Given two types of nodes, analysts create connections between them by clicking on one type of nodes and dragging the mouse to the other type of nodes in the network schema view (Figure 4(b)). This action draws an edge between the two that takes effect when the mouse button is released. To designate a quantitative column as edge weights, analysts drag and drop the column over the edge representation in the network schema view. Ploceus supports slicing and dicing for up to 2 dimensions, designated as the horizontal and vertical axes in the visualization. Analysts specify the orientation of the slices (horizontal or vertical) by dropping columns to the appropriate shelf (Figure 4(c)). These static figures give a basic idea of the interactivity of the interface, but we refer readers to the accompanying video for a more complete and richer view of the direct manipulation.

Analysts specify aggregation and projection, two transformative operations on existing networks, using dialog interaction rather than drag and drop. Currently Ploceus supports three types of aggregation operations: proximity grouping, binning and pivoting (Figure 5(a)). Analysts choose the type of aggregation through radio buttons. Depending on the properties of nodes selected, some operations may not be applicable. For example, when nodes have no attributes, pivoting does not make sense. To specify projection, analysts indicate through combo boxes the types of nodes to be projected (Figure 5(b)). Both dialogs offer previews of how the network will appear after the transformation, so that analysts can have a feel of the consequences of their actions.

Whenever analysts perform an operation, the network view provides immediate feedback in the form of a node-link visualization of the current network (Figure 2). Analysts can interactively add selected nodes and edges to the visualization through a search query field on the top right corner of the system toolbar (Figure 2). Analysts can also switch to a list-based view where different types of nodes are displayed in lists and the nodes are sorted by analytical metrics such as centrality. When the size of the network exceeds a threshold (currently defined as 450 nodes), to avoid screen clutter and low system performance, the node-link visualization will randomly sample and show a subpart of the network; the list-based visualization still shows the entire network.

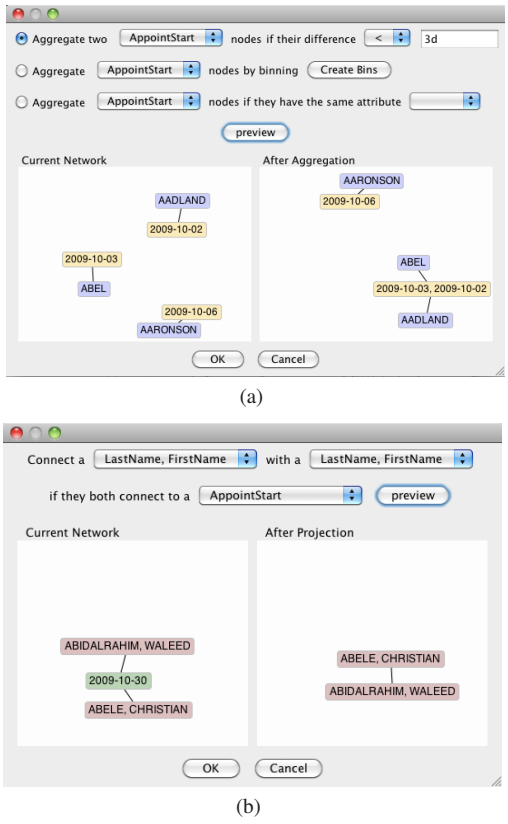


Figure 5: Dialogs for specifying a) aggregation and b) projection.

When slice 'n dice dimensions are specified, Ploceus shows a grid containing multiple small networks in the form of node-link visualizations only with brushing support (e.g. Figure 8). If the dimension used contains large number of categorical values, the large number of subnetworks can lead to usability and performance problems. This is one design issue that we would like to investigate in future work.

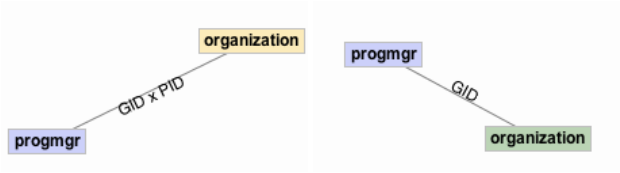
2.3 Edge Semantics and Construction Strategies

With such a set of diverse operations, it is important for analysts to correctly interpret the edge semantics in the networks created. When a network is created from a single table, the interpretation is usually straight-forward: e.g. connecting a visitor to a location indicates a visiting relationship, and the edge weight means frequency of visit. When these two types of nodes are from different tables, how the connections are constructed will affect the numerical weights assigned to the edges and how the edges are interpreted. For example, we can directly connect Program Manager nodes from Table 3(a) and Org nodes from Table 3(b), and the meaning of connection is that of managers granting awards to organizations. The exact meaning of the edge weight, however, is more subtle. Ploceus will determine that Table 3(c) already defines an explicit network relationship of Researcher x Grant (or GID x PID). This relationship is used to create edges, and as a result, the edges between program managers and organizations will have the semantics of ProgramManager – GID x PID – Org. The edge between Sylvia Spengler and University of California Irvine, for example, will have a weight of 3, indicating that she has awarded grants to researchers from this organization three times (to Sharad Mehrotra once and to Padhraic Smyth twice). That is, both the number of researchers per grant and the number of grants will have an impact on the edge weight.

This weight however may not be at the right level of abstraction to the analyst, as Sharad Mehrotra and Padhraic Smyth have collabo-

rated on a grant, and the program manager has in fact only awarded two grants to the organization. To let the weight reflect the number of unique grants awarded by the program manager to the organization only, we can connect Program Manager and GID explicitly first, then connect GID with Orgs. We then do a projection by connecting a Program Manager with an Org if they both connect to the same GID. The weight assigned to the edge between Sylvia Spengler and University of California Irvine will then be 2, indicating two grants.

These subtleties of edge construction reinforce that we can create connections between nodes with great flexibility and rich semantics. A program manager and an organization, for example, can be connected by the grants awarded by the manager to the organization, by the frequency of awards to researchers from this organization, or by the researchers from the organization who receive grants from the manager. This power comes with the requirement, however, of knowing the right operations to create the desired semantics. To help analysts keep track of what they are doing when connecting nodes from different tables, Ploceus labels the edge representation in the network schema view, indicating the semantics of the edges. Figure 6(a) shows the label for the first case and Figure 6(b) shows the label for the second case discussed in this section.



(a) (b)

Figure 6: Edge semantics labels in the network schema view

2.4 Visualization Management and Work Flow

Another consequence of providing a variety of construction and transformation operations is that it is now easy to generate a large number of distinct networks. Managing the networks thus becomes an important issue in the design of the user interface. In Ploceus, every network generated is associated with a tab. Analysts can generate new blank networks through the toolbar "New Network" button, and closing a tab deletes the network. Within each tab, analysts can switch between a node-link visualization and a list-based visualization; they can also tile these two visualizations side by side.

In the case of slicing and dicing, analysts can right click on any of the subnetwork and choose "Analyze in detail" in the pop-up menu. Ploceus will display the chosen subnetwork in a new tab, where analysts can examine it more closely and change the representation to list-based visualization. In this newly created tab, Ploceus remembers the specific slice 'n dice dimension values associated with the subnetwork, so analysts can choose to delete the network while keeping the slice 'n dice values for further exploration of alternative networks from the same perspective. Whenever a new network is created or deleted, or an existing network is transformed, the network schema view will update accordingly to reflect the schema of the network in the currently active tab. Saving and reloading networks are yet to be implemented.

3 Scenario: Analyzing Cross-Institution Research Efforts

To illustrate how to use the direct manipulation interface in conjunction with the visualization and computational capabilities provided by Ploceus for fast analytical insights, we present an example analysis in this section. For a more interactive and complete view of the analytic process, we refer readers to the accompanying video.

In this scenario we examine the research grants awarded by the NSF in the Information & Intelligent Systems division from 2000 to 2003. A subset of the data is presented in Table 3. It is a long-standing policy of NSF to encourage inter-institution research collaborations, and it would be of interest to understand the structure of collaboration networks at an organizational level. In particular, researchers from which organizations tend to collaborate with colleagues from other institutions? What factors might have influenced the collaborations?

The data set specifies an explicit 2-mode network at the actor level (PIs/co-PIs with grants). To construct a network at the organizational level, we drag and drop the organization column from the Person table and the GID column from the Grant table to the network schema view, and connect these two types of nodes. Immediately we have a network showing the connections between organizations and the grants they have received. To establish a direct linkage between organizations, we perform a projection on the GID nodes. Since we are only interested in organizations that have collaborated with at least one other organization, we filter out the organization nodes whose degree is 0. The network shown in Figure 7 results. We can see that the network is fairly well connected, with a few very small clusters detached from the main network. This indicates that the collaboration over the years is not segregated in isolated clusters, which is a positive sign. Switching to a list-based view and ranking the organizations by degrees, we see that Stanford University, University of California Berkeley, University of Washington, Columbia University and Georgia Tech are the top 5 cross-institution collaborators (please refer to the video). It is also interesting to note that Georgia Tech is the only one in the top 5 that has not collaborated with the other four organizations in the top 5.

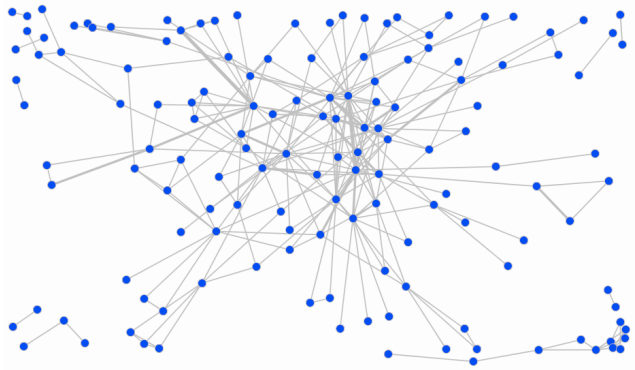


Figure 7: Collaboration between organizations on NSF IIS grants, 2000-2003

We can continue to explore the collaboration patterns of individual organizations, but to get a more systematic view of the structure of this network first, it may make sense to slice and dice it by both the year and the amount of the award. Assuming that we have defined how the amount dimension should be aggregated into categories, this gives us the network matrix in Figure 8. The visualization here seems to conspicuously refute our intuition about the relationships between grant size and collaboration: we would expect there would be less collaboration on small grants and more on larger grants. The visualization tells us instead that medium-sized grants seem to attract the least collaborations, and this observation is fairly consistent over the four years. Considering that there were 972 small grants awarded in this period compared with 159 medium grants and 133 large grants (shown in the shelf labels), however, the sheer number of small grants might just be the main reason that increases the chance of cross-institution collaborations. Upon closer examination, we can see that grant size does also play a part in shaping the structure of collaboration networks. For small grants, two-organization collaboration is very typical, while for large grants, such collaboration patterns are much less common.

In particular, there is a high level of collaboration occurring in large grants awarded in 2003.

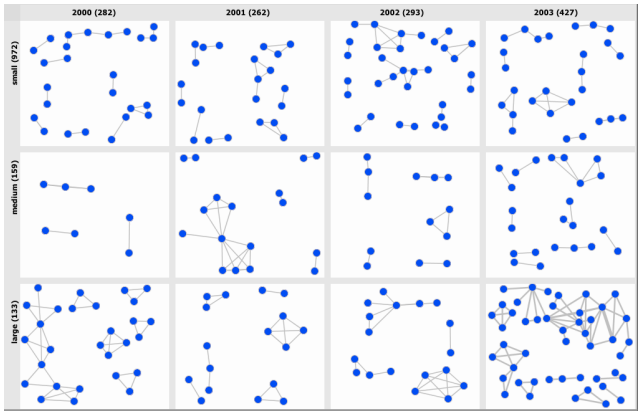


Figure 8: Collaboration between organizations on NSF IIS grants, broken down by year and amount

To investigate further, we right click in the 2003-large grant cell and choose "Analyze in detail" to open a new tab showing that subnetwork for closer analysis. We can see that University of Colorado at Boulder (CU Boulder for short) occupies an important position in this subnetwork where it connects multiple local clusters (Figure 9). This observation is confirmed after running the computational analysis, where CU Boulder has the highest betweenness centrality score, indicating that it is linking many organizations that are otherwise not linked. One reason for this is that CU Boulder has collaborated on quite a few different large grants with different organizations in 2003. To see the grants it has received as well as the collaborating institutions for each grant, we clear the current subnetwork while keeping the 2003-large grant slice specification, and construct an organization-name-title network, connecting organizations with the researchers who are connected with the grants they receive. We see the specific researchers from this school as well as the three large grants they have worked on: emotion in speech, tangible media and semantic interpretation (Figure 10).

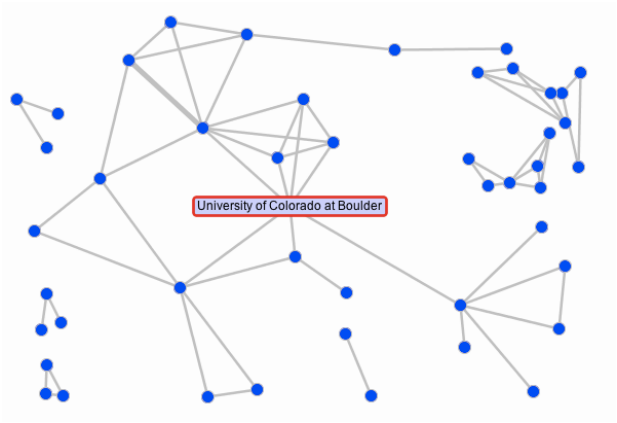


Figure 9: CU Boulder is an important actor in the 2003-large grant collaboration network

To look further at the role of program managers in the collaboration dynamics, we now go back to the previous tab and replace the date slices with program manager slices. Noting that William Bainbridge, Maria Zemankova, and Ephraim Glinert are the top 3 grant awarding managers, we find that a significant portion of their

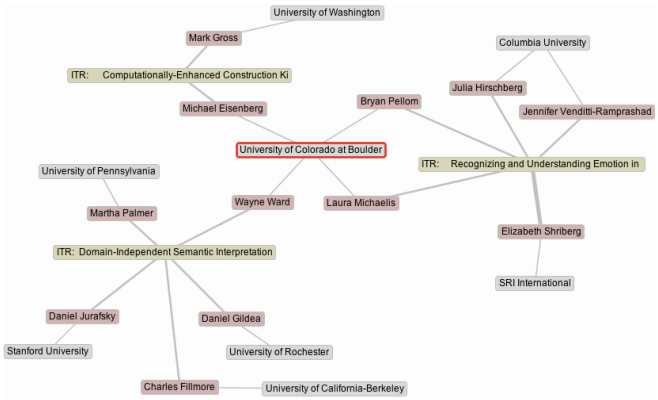


Figure 10: Large grants received by CU Boulder and other institutions in conjunction in 2003

grants is small grants. After filtering out non-collaborating institutions, we find that grants awarded by them do not particularly show greater activities of collaboration (Figure 11). It is also obvious from the visualization that Ephraim Glinert has awarded a number of grants to groups of 4 institutions (visualized in the form of tetrahedra), and Stephen Griffin awarded one grant to a group of 5 collaborating institutions (in the form of a pentahedron). Such patterns, some of which are highlighted in the figure, are not seen in grants awarded by other program managers.

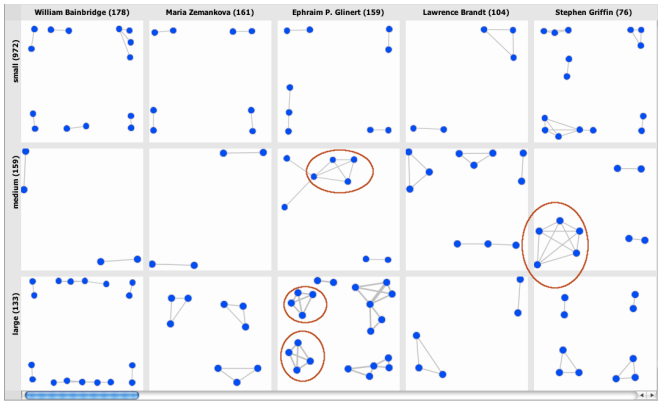


Figure 11: Collaboration between organizations on NSF IIS grants, broken down by program manager and amount

4 Computing Connections

The logic underlying Ploceus is built on top of a formal framework that systematically specifies how to compute edge connections and assign edge weights. We will present a detailed treatment of the framework at the level of formal definitions and proofs in a partner paper. Here, we provide an overview of the framework.

4.1 Approach and assumptions

Analysts that organize data into structured rows and columns in tables are implicitly declaring relationships between data elements. When data elements appear in the same column, they usually belong to the same type (e.g. 142 and 16 are both GroupSize in Table 1). When data elements appear in the same row, they are usually semantically related, and the specific semantics depend on the context. When Aarnio, Alicia and OEOB appear in a single row of the White House visit logs, this co-occurrence can be interpreted as a visiting relationship between two entities: the person Alicia Aarnio visited the Old Executive Office Building (OEOB). When

Data Mining of Digital Behavior and 2241750 appear in the same row of the NSF grant data, this co-occurrence can be interpreted as a description of an entity in terms of an attribute: the amount of the grant is $2241750.

Our approach leverages this simple observation that *the meaning of row-based co-occurrence is context-sensitive*. It is thus possible to propose a co-occurrence based formal framework which specifies the construction and transformation of networks, where the meaning of the graphs created will be subject to users' interpretation. Co-occurrence is undirected: when A co-occurs in a row with B, B co-occurs with A too. The meaning of connections derived from co-occurrence, however, can be interpreted as having a sense of direction. For example, the meaning of the connection between a person and a location is a visit *by* the person *to* the location.

We base our formal framework on the relational model [17] used pervasively in database theories, with basic relational algebraic operators such as selection (σ), projection (π), join ($\bowtie$) and aggregation ($\mathscr{F}$) [18]. We make the following three assumptions:

- Each row in a table has a unique identifier.
- Each value in the table cells is *atomic*, i.e. the value can be classified as nominal, quantitative and ordinal, and the value cannot be decomposed into meaningful smaller units.
- We only focus on creating networks in which there are no edges connecting one node to itself.

4.2 First-order Graphs

The entire formal framework is built on the fundamental notion of a *first-order graph* and transformative operations on the graph. First-order graphs are the simplest graphs or networks we can construct where each node and edge is constructed from one (1) single row only. In relational model terms, a row is a *tuple*, where one or more cell values in that row form a *subtuple*, and a table is a *relation*. When all the data needed for graph construction are present in a single table, for any given row in a table, there are two main ways to construct a node from it. We can create a node such that its label is a subtuple (e.g. the node label is "Smith, John", or a function of a subtuple (e.g. taking Size as the argument, and returns "large group" as the node label if the group size is above 50, and "small group" otherwise). In a similar way, we can assign an attribute to a node based on a subtuple or a function of subtuple.

It is thus a basic idea that in the translation from a table to a graph, if the construction of a node results from only a single row of the table, the node is a *first-order* node. Two first-order nodes can have the same labels and attributes, as there may be rows containing identical values for selected table dimensions. First-order nodes are created using the relational projection operator.

Here we introduce two important concepts: **locale** of a node and **basis** of an edge in order to compute connections consistently when multiple tables and graph transformation are involved. The locale of a node refers to the set of tuples from which the node is constructed; the basis of an edge refers to the set of relational elements (tuples or graph nodes) which are jointly shared by the locales of two nodes. In actual implementation, the comparison of locale and computing of edges are realized using the relational selection and projection operators. The *weight* of an edge will be the *cardinality of its basis*.

In first-order graphs for example, the locale of a node will just contain one element: the tuple from which the node is created. As mentioned earlier, our formalism focuses on establishing relationships based on co-occurrence in rows. Two first-order nodes are thus connected if they share the same locale. Formally speaking,

$$\text{If } \exists t,\ locale(n_1) = locale(n_2) = t, \text{ then } e(n_1, n_2).$$

Our framework considers two possible cases when first-order nodes and edges are constructed from multiple tables. First, we can create two sets of first-order nodes, each constructed from a single table only, and the edges between the nodes are created by linking two tables. The notion of co-occurrence is then no longer limited to one tuple in a relation, but is extended to include two or more tuples in multiple relations through a join condition specified by the analyst. Formally,

$$\begin{aligned}&\text{Given } locale(n_1) = R_1.t_i \wedge locale(n_2) = R_2.t_j,\\ &\text{If } (R_1.t_i \cup R_2.t_j) \in (R_1 \bowtie_\theta R_2), \text{ then } e(n_1, n_2).\\ &basis(n_1, n_2) = \{(R_1.t_i, R_2.t_j)\}.\end{aligned}$$

In the second and more complex case, a set of first-order nodes can be constructed such that their labels come from one table, and their attributes come from another table. We do not allow constructing node labels from multiple relations in our formalism for the purpose of simplicity. The type of join used here in constructing first-order nodes will be a left-outer-join [18] because we want to preserve all the node labels even when there are no matching attributes. The locale of the nodes is determined by the table from which the labels are constructed only.

4.3 Higher-order Graphs: Transformation

First-order graphs often are not at the right level of abstraction intended for exploration and analysis. For example, there may be nodes with identical labels that refer to the same entity. In Section 2.1 we introduced three transformative operations: aggregation, projection and edge weighting. We also mentioned that Ploceus aggregates nodes by labels and attributes automatically. Our formal framework specifies how these transformations affect the edges based on the notion of a locale introduced in the previous section. In aggregation, for example, assuming the analysts have specified a function of aggregating nodes, the newly produced nodes will inherit the locales of the nodes being aggregated:

$$locale(n') = locale(n_1) \cup \ldots \cup locale(n_j).$$

Two new nodes will be connected if the intersection of their locales is not empty:

$$basis(n'_1, n'_2) = locale(n'_1) \cap locale(n'_2) \neq \emptyset$$

For projection on a two-mode graph with two types of nodes N and M, for example, two nodes $n_1, n_2 \in N$ are connected if they have at least one neighbor in common in M:

$$\exists m \in M,\ e(n_1, m) \in E \ \&\ e(n_2, m) \in E \Rightarrow e(n_1, n_2)$$

According to this definition, the basis of an edge is no longer a set of tuples, but a set of nodes:

$$basis(n_1, n_2) = \{m \in M \mid e(n_1, m) \in E \ \&\ e(n_2, m) \in E\}$$

Slicing and dicing are operations at a global level using dimensions that are orthogonal to those used in network construction. In our framework, the dimensions used in slicing and dicing serve as query conditions when nodes and edges are created through relational selection and projection operators. Ploceus currently infers equi-join conditions by analyzing foreign key constraints between tables through a Dijkstra shortest-path algorithm. This feature is useful when we are creating graphs from multiple tables, or the slicing/dicing dimensions and node/edge construction columns are from different tables. When multiple equi-joins are possible, or the desired join condition is not based on equality, analysts can specify the condition through a dialog window.

4.4 Expressive Power and Limitations

In working with sample datasets we have already identified situations that point to potential limitations of the current framework. For example, if we want to create a network where two organizations are connected if they have collaborated on more than two grants within the past five years, the set of operations described in Section 2.1 is not sufficient to express such semantics of conditional connectivity.

Further work is required to understand the expressive power and limitations of this framework. Relational algebra, an established framework, is proven to be equivalent to first-order logic, and the expressive power of first-order logic is well understood [18]. In relational algebra, a set of primitive operators serves as building blocks for more complex operators. Since we are investigating a new domain here, it remains to be seen if the set of operations can serve as primitives for graph construction and if any additional operations need to be included for completeness.

5 Related Work

A number of systems in the form of toolkits [22, 27] or executables [1, 2, 3, 5, 10, 12, 28] are available for analyzing a given graph. These systems vary in features and provide visualizations, computational metrics or both. NodeTrix [23] explores how these different network representations can be integrated for the same underlying graph data. ManyNets [19] looks at visually exploring multiple networks. PivotGraph [35] provides attribute-based transformation of multivariate graphs. Creating and transforming network semantics from data tables are not the main focus of these systems.

NodeXL [21] integrates with Microsoft Excel to enable users to easily import, transform, visualize and analyze network data. The data transformation power, however, is primarily provided by Excel and decoupled from visual exploration.

Systems such as Table Lens [29], FOCUS [32], InfoZoom [31], Polaris [34] and Tableau [4] visualize tabular data in the form of line charts, bar charts, scatter plots or space filling cells for analyzing distribution pattern and frequency aggregation. None of these systems pays special attention to the potential of imposing user-defined relationships between attribute values in the form of networks. Our motivation behind designing Ploceus does resonate with the approaches taken by Polaris and Tableau, which advocate the need for analysts to rapidly change the data they are viewing and how the data are visualized, as well as the need to integrate data transformation and visual abstraction in a seamless process.

Jigsaw [33] builds the semantics of relationships into the system design based on a simple assumption: entities are identified purely lexically, and entities appearing in the same documents are connected. This approach, originally designed for unstructured text documents, can be extended to tabular data such as spreadsheets: one row in a table is equivalent to the notion of a document. The co-occurrence based definition allows flexible explorations of entity relationships without having to explicitly specify the nodes and edges, but since the fundamental connection model is centered around documents/rows, the connections between table columns are less direct. Jigsaw also has limited data transformation support due to its indiscrimination between nominal and quantitative entities.

Ploceus focuses on extracting trees and graphs from data tables, and variants of this idea have been explored in prior work. The attribute relationship graphs approach establishes direct connections between table column values and integrates the graph with cross-filtered views [36]. The need for retrieving and publishing selected information on the web leads to work that models databases as virtual graphs [20] and provides XML document interfaces of relational data for web applications [14]. The Grammar of Graphics discusses an algebraic framework for mapping tables to directed trees [37]. Commercial systems such as Touchgraph Navigator [9] and Centrifuge [6] provide interface for creating attribute relationship graphs from data tables. While Ploceus is not the first system that investigates the connection between data tables and graphs, our approach differs from existing work in two ways: first, we offer a comprehensive construction and transformation framework that integrates diverse operations in a flexible yet systematic manner; and secondly, the design of the system tightly couples data transformation with visual exploration.

Finally, in the area of data mining and knowledge discovery, there is a body of literature dealing with mining insights from graph data. While most of the work focuses on automatic mining techniques, the graph OLAP [15] is of particular relevance to our work. Inspired by the traditional relational model based OLAP, the graph OLAP proposes a framework for performing multi-dimensional and multi-level operations on a graph. In particular, the approach specifies an informational dimension based OLAP, which corresponds to our notion of slice 'n dice, and topological dimension based OLAP, which corresponds to our notion of aggregation. Our work focuses more on the visual and interactive part of the mining process, and our framework is arguably broader because we do not require the data to contain explicit network semantics.

6 Implementation

Ploceus is built entirely in Java on the NetBeans Rich Client Platform [8]. It utilizes two major external toolkits and libraries: H2 [7] as the underlying database for relational algebraic queries, and JUNG [27] as the graph visualization and computational metrics library. All the operations supported by Ploceus are performed in real time: simple operations such as adding nodes and creating connections are realized through SQL queries and are scalable for up to tens of thousands of rows without significant delay. More complex operations such as projection and statistical metrics computation are more computationally expensive and the performance can be affected with large data sets. Every subnetwork in slicing and dicing is created through a separate thread, and the performance bottleneck is at the concurrent handling of SQL queries by the underlying H2 database. Future work includes optimization of the implementation of operations.

7 Conclusion and Future Work

In this paper, we have focused on the system design and formal framework aspects of performing network-based visual analysis, and argued our approach provides new capabilities beyond existing work. Our contributions include the following:

- A conceptual framework specifying possible operations for constructing and transforming networks from multivariate tabular data.
- A specification of the operations based on the relational model and an implementation of the framework in relational algebra.
- The design and implementation of a system based on the framework, which integrates data manipulation with visual exploration processes.
- A discussion of the nature of high-level tasks in network-based visual analysis that may have implications for future work on visual analytics.

This research lays the foundation for further investigations. First, there are certain features we would like to add, such as pinpointing specific data rows/columns from visualizations and integrating analytic techniques on data tables such as log-linear modeling on top of the network analysis techniques presented here. As mentioned in Section 4.4, it is worthwhile to understand the expressive power and limitations of our framework in greater detail, and perhaps to examine how this framework can be applied and extended for compound graphs. We would also like to put the system into real users' hands, and gather feedback to refine our framework and design.

Acknowledgements

This work is supported by the National Science Foundation under award IIS-0915788 and the VACCINE Center, a Department of Homeland Security's Center of Excellence in Command, Control and Interoperability.

REFERENCES

[1] Gephi, an open source graph visualization and manipulation software. http://gephi.org, Aug. 2010.

[2] NetMiner - social network analysis software. http://www.netminer.com, Aug. 2010.

[3] Pajek - program for large network analysis. http://pajek.imfm.si/doku.php, Aug. 2010.

[4] Tableau software. http://www.tableausoftware.com/, Aug. 2010.

[5] UCINET. http://www.analytictech.com/ucinet/, Aug. 2010.

[6] Centrifuge systems. http://centrifugesystems.com/, June 2011.

[7] H2 database engine. http://www.h2database.com/html/main.html, Mar. 2011.

[8] NetBeans Rich-Client platform development. http://netbeans.org/features/platform/, Mar. 2011.

[9] TouchGraph navigator: Graph visualization and social network analysis software. http://touchgraph.com/navigator, Mar. 2011.

[10] E. Adar. GUESS: a language and interface for graph exploration. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, 2006.

[11] R. A. Amar and J. T. Stasko. Knowledge precepts for design and evaluation of information visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):432–442, 2005.

[12] D. Auber. Tulip: a huge graph visualization framework. *Graph Drawing Software, Mathematics and Visualization*, 2003.

[13] S. Bagui and R. Earp. *Database Design Using Entity-Relationship Diagrams*. Auerbach Publications, 1 edition, 2003.

[14] P. Bohannon, H. F. Korth, and P. P. S. Narayan. The table and the tree: On-line access to relational data through virtual XML documents. In *Proc. of WebDB*, 2001.

[15] C. Chen, X. Yan, F. Zhu, J. Han, and P. S. Yu. Graph OLAP: a multi-dimensional framework for graph data analysis. *Knowledge and Information Systems*, 21:41–63, Oct. 2009. ACM ID: 1669197.

[16] P. P. Chen. The entity-relationship model – toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[17] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):387, 1970.

[18] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Addison Wesley, 6th edition, 2011.

[19] M. Freire, C. Plaisant, B. Shneiderman, and J. Golbeck. ManyNets: an interface for multiple network analysis and visualization. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 213–222, 2010.

[20] S. Guéhis, P. Rigaux, and E. Waller. Data-driven publication of relational databases. In *10th International Database Engineering and Applications Symposium*, pages 267–272, 2006.

[21] D. Hansen, B. Shneiderman, and M. A. Smith. *Analyzing Social Media Networks with NodeXL: Insights from a Connected World*. Morgan Kaufmann, Sept. 2010.

[22] J. Heer, S. K. Card, and J. A. Landay. prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, 2005.

[23] N. Henry, J. Fekete, and M. J. McGuffin. NodeTrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007.

[24] M. Latapy, C. Magnien, and N. D. Vecchio. Basic notions for the analysis of large two-mode networks. *Social Networks*, 30(1):31–48, 2008.

[25] B. Lee, C. Plaisant, C. S. Parr, J. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pages 1–5, Venice, Italy, 2006. ACM.

[26] Z. Liu and J. Stasko. Mental models, visual reasoning and interaction in information visualization: A top-down perspective. *IEEE Transactions on Visualization and Computer Graphics*, 16(6), 2010.

[27] J. O'Madadhain, D. Fisher, S. White, and Y. Boey. JUNG: Java Universal Network/Graph Framework. Technical Report Technical Report UCI-ICS 03-17, 2003.

[28] A. Perer and B. Shneiderman. Balancing systematic and flexible exploration of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):693–700, 2006.

[29] R. Rao and S. K. Card. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, pages 318–322, Boston, Massachusetts, United States, 1994. ACM.

[30] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):733–740, 2006.

[31] M. Spenke and C. Beilken. InfoZoom: Analysing Formula One racing results with an interactive data mining and visualisation tool. In *Proceedings of 2nd International Conference on Data Mining*, pages 455–464, Cambridge University, UK, 2000.

[32] M. Spenke, C. Beilken, and T. Berlage. FOCUS: the interactive table for product comparison and selection. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 41–50, 1996.

[33] J. Stasko, C. Görg, and Z. Liu. Jigsaw: supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2):118–132, 2008.

[34] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, pages 52–65, 2002.

[35] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 811–819. ACM New York, NY, USA, 2006.

[36] C. Weaver. Multidimensional data dissection using attribute relationship graphs. In *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pages 75–82, 2010.

[37] L. Wilkinson. *The grammar of graphics*. Springer Verlag, 2005.