Info Vis

Article

Information Visualization
2014, Vol 13(1) 59–89
© The Author(s) 2013
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1473871613488591
ivi.sagepub.com
SAGE

Ploceus: Modeling, visualizing, and analyzing tabular data as networks

Zhicheng Liu[1], Shamkant B Navathe[2] and John T Stasko[2]

Abstract
Tabular data are pervasive. Although tables often describe multivariate data without explicit definitions of a network, it may be advantageous to explore the data by modeling it as a graph or network for analysis. Even when a given table design specifies a network structure, 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 multidimensional and multilevel 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 through immediate feedback mechanisms. We report our findings on the learnability and usability of Ploceus and propose a model of user actions in visualization construction using Ploceus.

Keywords
Data transformations, graph and network visualization, multivariate data, interaction design, exploratory data analysis

Introduction

Network visualizations, often in the form of node-link diagrams, are an effective means to understand the 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. 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 multidimensional and multilevel 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 in the following, tabular data may or may not contain explicit specification of nodes and edges in a graph, and its multivariate nature implies the need for dynamic network modeling for greater analytic power.

Forms of tabular data

Tabular data come in many forms, each unique in its schematic and semantic structures 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 article.

[1]Stanford University, Stanford, CA, USA
[2]Georgia Institute of Technology, Atlanta, GA, USA

Corresponding author:
Zhicheng Liu, 465 Wilton Ave, Palo Alto, CA 94306, USA.
Email: zcliu@cs.stanford.edu

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

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

OEOB: Old Executive Office Building; VA: Visitor Access; AL: Agency Liaison; WH: White House; OEOB: Old Executive Office Building; POTUS: President of the US.

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

(a) Employee

ID	FName	LName	Bdate	Dpt
1	John	Smith	10 Jan 65	2
2	Franklin	Wong	9 Apr 52	3
3	Jennifer	Wallace	23 Oct 70	3
4	Ahmad	Jabbar	2 Nov 45	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

Single tables are represented as 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 reference to other entities. In multivariate data, explicit definition of a graph structure is typically absent.

Multiple linked tables are often stored in relational databases, although the same tables can also be described in spreadsheets. In a relational database, the entity–relationship (ER) model[1] 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 work for only one department. *One-to-many* relationships are typically captured by foreign keys in a relational database.[2] In this case, Dpt in the EMPLOYEE table is a foreign key, referencing the DEPARTMENT table.

Another type of relationship in the ER model is the *many-to-many* relationship, and it is captured by a separate relationship table.[2] For example, Table 3(a) represents selected facts about research grants awarded by the National Science Foundation (NSF) in the Information & Intelligent Systems (IIS) 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 also involve multiple researchers.

These tabular data in multiple linked tables are essentially *attributed graphs*. 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 online analytical processing (OLAP) database, unlike spreadsheets and relational databases, is not built for low-level transactional operations such as insertion and update, but for retrieval, querying, and 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

Table 3. Tables describing researchers and the grants they receive.

(a) Grant

GID	Title	Program	Program Manager	Amount
1	Data mining of digital behavior	Statistics	Sylvia Spengler	2,241,750
2	Real-time capture, management and reconstruction of spatiotemporal events	Information Technology Research	Maria Zemankova	430,000
3	Statistical data mining of time-dependent data with applications in geoscience and biology	ITR for National Priorities	Sylvia Spengler	566,644

(b) Person

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

(c) Work-on

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

ITR: Information Technology Research; PI: Principle Investigator; CoPI: co-Principle Investigator.

necessarily suitable for network-based analysis because it focuses only on inherent relationships between entity attributes and assumes different entities are mutually independent.[3] As a result, the OLAP framework is not directly relevant for our purpose, and in this article, we focus on spreadsheets and databases, which provide a basis for an alternative network-centric framework.

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, the concept of an entity is often multilevel and 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.[4] Second, 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? That is, 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

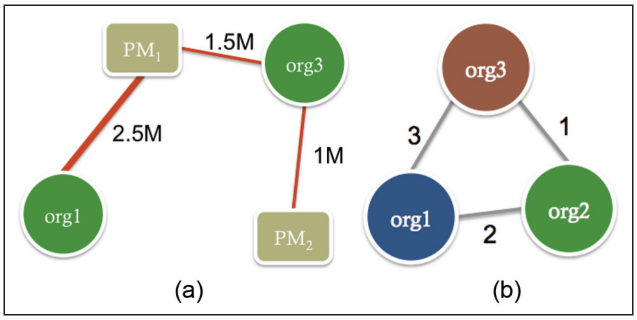


Figure 1. Visual models for answering questions on the NSF dataset. (a): a sample network where a program manager is connected to an organization if the manager has given grants to researchers from the organization; edge weight indicates amount of grants, (b): a sample network where an organization is connected to another organization if they have received grants together; edge weight indicates number of grants.
NSF: National Science Foundation.

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 Figure 1 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.

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 have two major characteristics. First, they cannot be answered satisfactorily by simple "yes" or "no" or precise quantification. Analysts can define metrics to measure "affiliation strength," for example, in the case of Q1, but such metrics are only meaningful at the level of specific manager–institution pairs. Network visualizations are useful to show global structures in the network. Second, these questions 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.

Amar and Stasko[5] considered answering such questions as performing *high-level* analysis tasks, which can be contrasted with *low-level* tasks[6,7] that 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.

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.[8] Illustrations of analysts' desired model based on their analytical questions are given in Figure 1(a) and (b). 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.

Objective and organization

With these considerations in mind, we present Ploceus (Ploceus is a kind of weaver bird that can build sophisticated nests), 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.[9–15] 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.

This article is an expanded and updated version of a paper presented at the VAST 2011 Conference.[16] In this version, we present research findings on additional issues related to network-based visual analysis in tabular data. More specifically, section "Visual encoding" explores automatic visual encoding of networks modeled from data tables. Sections "Extending to one-mode networks" and "Extending to one-mode graphs" extend the scope of our formal framework and the system to support the construction of one-mode networks from reflexive relational tables. Section "User evaluation" reports our findings on the learnability and usability of Ploceus from a qualitative user study. We identify potential roadblocks in network modeling and propose a model of user interaction with Ploceus.

Related work

Visualizing multidimensional data

Systems such as DEVise,[17] Table Lens,[18] FOCUS,[19] InfoZoom,[20] Polaris,[21] and Tableau[22] visualize tabular data in the form of line charts, bar charts, scatterplots, 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[23] 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.

Network visualization and analysis

A number of systems in the form of toolkits[24,25] or executables[9–12,15,26,27] are available for analyzing a given graph. These systems vary in features and provide visualizations, computational metrics, or both. NodeTrix[14] explores how these different network representations can be integrated for the same underlying graph data. ManyNets[28] looks at visually exploring multiple networks. PivotGraph[29] provides attribute-based transformation of multivariate graphs. Creating and transforming network semantics from data tables are not the main focus of these systems.

NodeXL[13] integrates with Microsoft Excel to enable users to easily import, transform, visualize, and analyze network data. NodeXL stores network data in multiple sheets representing nodes and edges, and users likely will need to be Excel experts to be able to transform the data.

Attribute relationship graphs

Ploceus focuses on extracting trees and graphs from data tables, and variants of this idea have been explored in prior study. The need for retrieving and publishing selected information on the web leads to work that models databases as virtual graphs[30] and provides Extensible Markup Language (XML) document interfaces of relational data for web applications.[31] The Grammar of Graphics discusses an algebraic framework for mapping tables to directed trees.[32] Weaver proposed a data transformation pipeline for attribute relationship graphs[33] and is perhaps the closest to our algebraic framework presented in section "Computing connections."

A number of systems appear to be close to Ploceus in terms of design goal and functionality, including CineGraph demonstrating the attribute relationship graphs approach,[33] two Commercial systems TouchGraph Navigator[34] and Centrifuge,[35] and the Orion system.[36] Weaver[33] distinguishes between attributed graphs (where an object is connected to its attributes) and attribute relationship graphs (where attributes are connected based on occurrence). This notion of attribute relationship graph lays the foundation of our study. Weaver's Cinegraph system supports many network modeling operations such as deriving attributes and slicing and integrates the generated network with cross-filtered views.[37] TouchGraph Navigator[34] and Centrifuge[35] provide interfaces for creating attribute relationship graphs from data tables. The Orion system,[36] published concurrently with our VAST paper, also supports the construction and transformation of network data.

While Ploceus is not the first system that investigates the connection between data tables and graphs, we distinguish our study from related systems in two ways. First, we offer a comprehensive construction and transformation framework that integrates diverse operations in a flexible yet systematic manner (section "Operations" describes these operations in detail). Table 4 summarizes the operations provided by Ploceus and the related systems. The same operation may be named differently in different systems, and we provide the terms used by these systems whenever possible. Due to the inaccessibility of the commercial software TouchGraph Navigator,[34] we cannot do a comprehensive assessment of its features and thus

Table 4. A comparison between different systems in terms of the network modeling operations provided.

	Ploceus	Centrifuge	ARG	Orion
Create nodes	✓	✓	✓ (group)	✓ (promote)
Add attributes	✓	×	×	×
Create connections	✓	✓	✓ (clique)	✓ (link)
Assign weights	✓	×	✓ (weight)	✓ (weight)
Project	✓	×	×	*
Aggregate—pivoting	✓	×	×	✓ (roll-up)
Aggregate—binning	✓	✓	×	×
Aggregate—proximity grouping	✓	×	×	×
Slice 'n dice	✓	✓	✓ (slice)	✓ (split)
Filter by value	*	×	✓ (drill)	✓ (filter)

omit it from the comparison. As is evident from the table, all the systems provide support for basic operations such as creating nodes and connections. Adding node attribute, pivoting, projecting, and proximity grouping are absent in one or more of the other systems. Due to different interface designs, sometimes, there is no direct one-to-one mapping between the operations in our framework and those in other systems, and we indicate such situations as "*" in Table 4. For example, in Orion, there is no "project" operation, but users can create one-mode networks by "promoting" a column to a table and connect the column to itself.[36] Similarly, in our framework, there is no explicit "filter by value" operation, but analysts can identify specific node values through the search function provided at the interface level.

Second, we take a human-centered perspective in interface design. While systems such as Orion and CineGraph provide similar modeling operations, their interfaces are significantly different from those of Ploceus. In our design considerations, we aim to expose each network modeling operation as a conceptually meaningful unit to the users and map each operation to an action or an interface element. In particular, we design and implement a network schema view based on the notion of "visualization schemas"[38] to enable analysts to construct a network by combining the modeling operations without the need for programming.

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 are integrated. 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.

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 "Computing connections."

- *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 can create the labels of the nodes 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 also supports adding columns as attributes from a different table. 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 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, for example, ProgramManager nodes from Table 3(a)

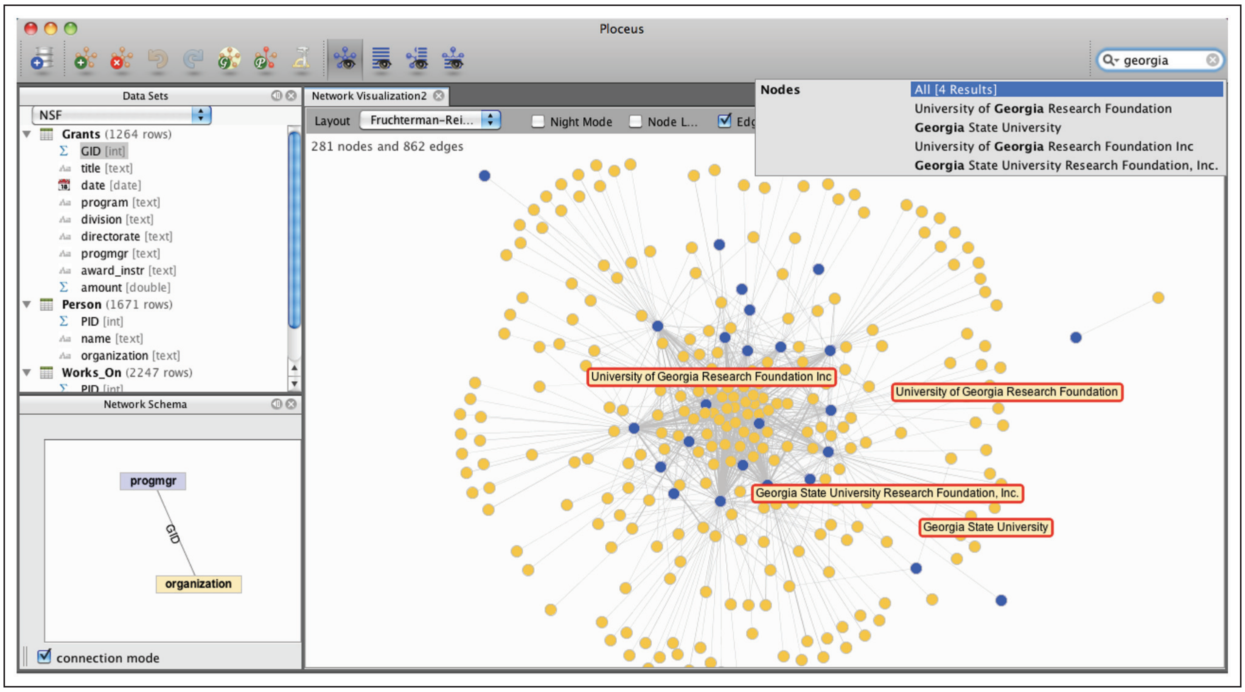


Figure 2. Ploceus system interface with a data management view on the top left, a network schema view on the bottom left, and a network visualization view on the right.

and Org nodes from Table 3(b). When multiple tables are involved, Ploceus tries to determine how the tables should be joined by analyzing the foreign key constraints between the tables using the Dijkstra shortest-path algorithm[39] (section "Higher-order graphs: transformation" provides technical details on this issue). 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 (sections "Edge semantics and construction strategies" and "Computing connections" 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.[40] 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[29] terms this operation *roll-up*. Given LName, FName nodes with the attribute AccessType, we can aggregate people nodes

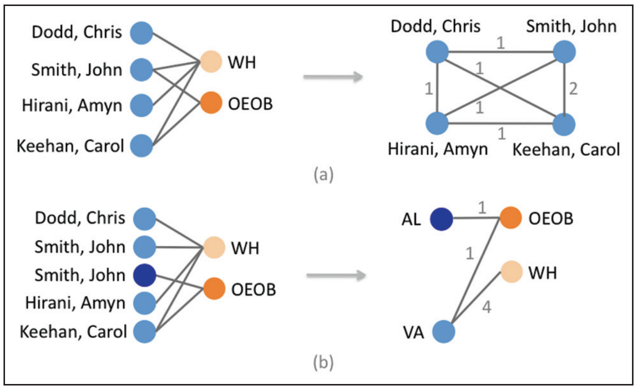


Figure 3. Project and pivot operations: (a) the visitor nodes do not have attributes and (b) the visitor nodes have attribute "Type" (with values "AL" and "VA") from the original data table.
OEOB: Old Executive Office Building.

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 $\leqslant$ 500 K, "medium" if 500 K < Amount $\leqslant$ 1200 K, and "large" if Amount > 1200 K.
- *Proximity grouping*. Group nodes in a pairwise 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 everyone is within 500 miles of 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, and then project employees on cities).
- *Slice 'n dice*. Divide a network into subnetworks based on selected columns. For example, given that we have constructed an 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, for example, for a *date* column: day $\rightarrow$ week $\rightarrow$ month $\rightarrow$ quarter $\rightarrow$ year.

We try to be comprehensive in choosing the relevant operations to be included based on three criteria: (1) The operation is indispensable for creating a basic network, (2) earlier related work shows the utility of the operation, and (3) the operation is considered useful based on our own experience in performing network-based visual analysis. Section "Expressive power" explores the issue of expressive power offered by these operations. 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-mentioned operations for more complete user experience in performing data transformation and visual exploration.

Design of direct manipulation interface

In designing Ploceus, we wanted to make the interface accessible for users who do not necessarily possess programming skills. To achieve this goal, it is desirable to reduce articulatory distance, that is, assuming the analysts want to perform some operations, what is an intuitive way for them to communicate the intent to the system.

One possible design is to integrate a visual interface with a scripting interface as done in GUESS[12]—the manipulation of graphs is in the form of commands. The advantage of this approach is that script languages are precise, expressive, and concise; the disadvantage of this approach is that analysts must understand basic programming concepts.

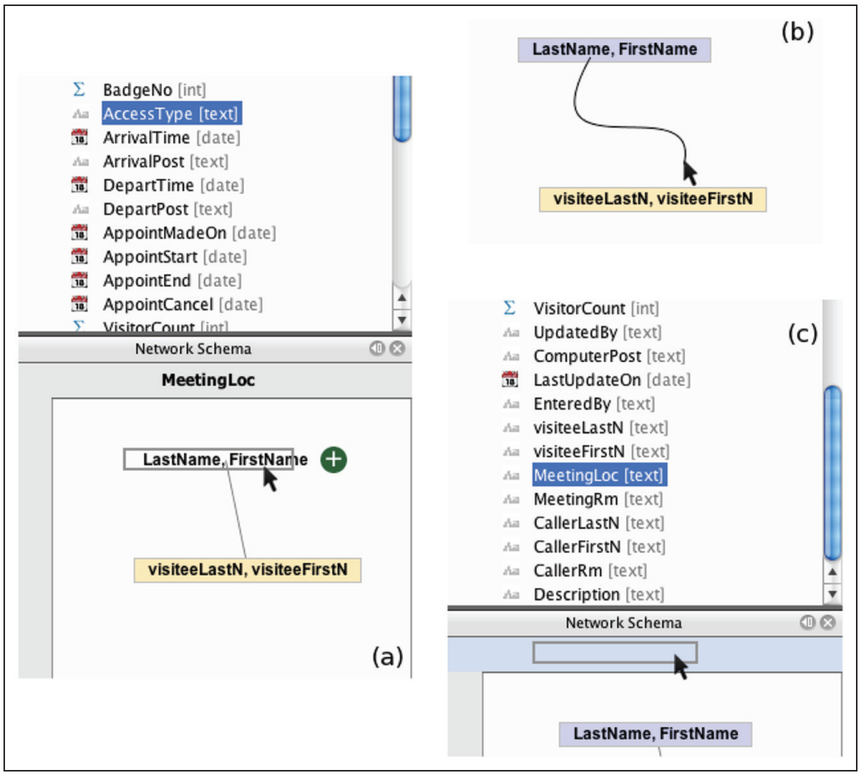


Figure 4. Direct manipulation interfaces for various operations: (a) add attributes, (b) create connections, and (c) slice 'n dice.

Another design alternative is programming by demonstration (PBD).[41] PBD typically uses a direct manipulation interface. For example, to create connections between LName, FName nodes and Loc nodes constructed from Table 1, assuming that we already have a visual representation of the existing nodes, we can use a click–drag–drop mouse gesture to connect a visitor node (e.g. "Dodd, Chris") and a location node (e.g. "WH"). After we have performed similar gestures two or three times, the system will figure out that our intention is to connect LName, FName nodes and Loc nodes and will perform the same operation automatically on the rest of the nodes.

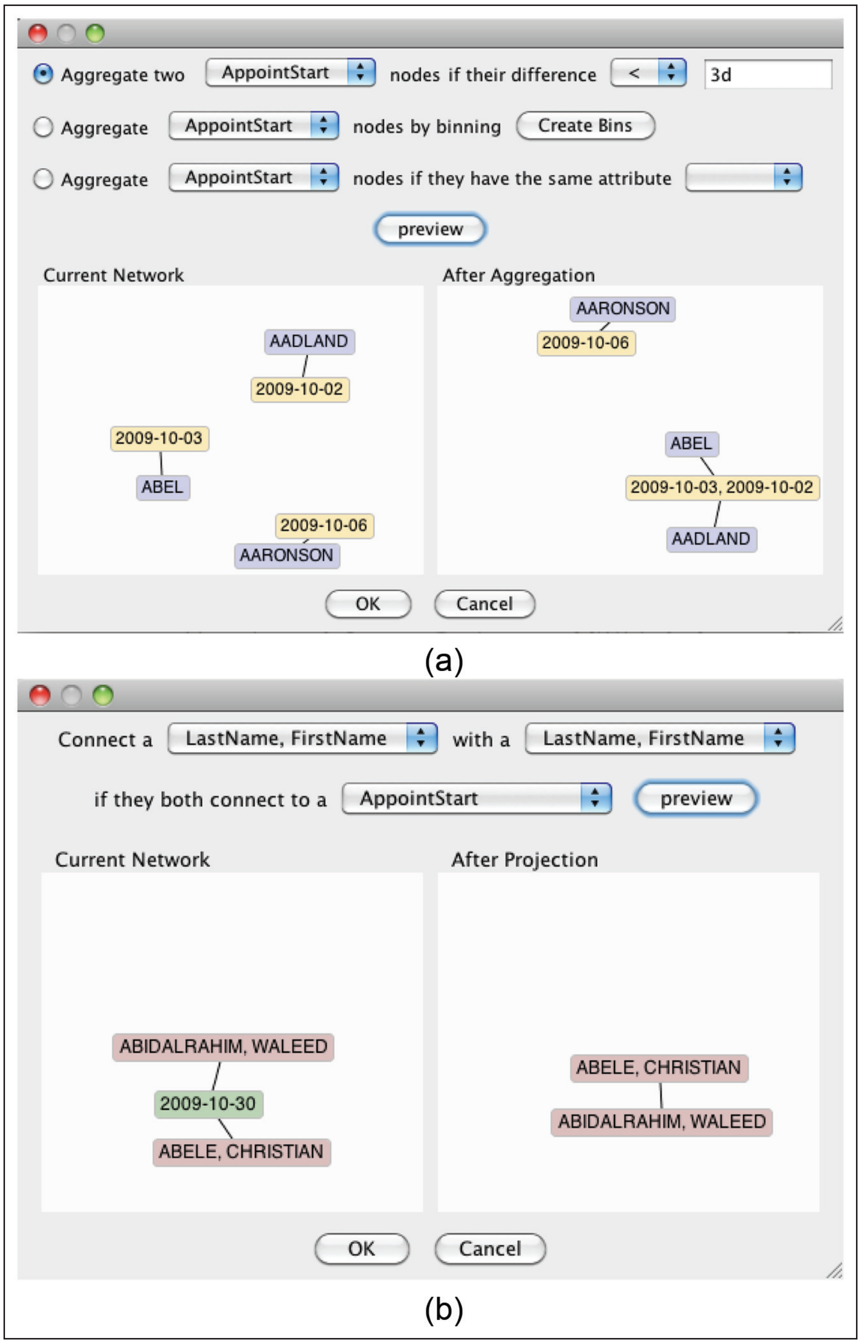
PBD arguably shortens articulatory distance when it works on a direct manipulation interface. As shown in the create-connection example, users perform the exemplary operation at the level of individual data items, and the system generalizes from the user interaction to a high level by connecting different types of nodes. This bottom-up design approach has some shortcomings for network modeling. Analysts need to know if an edge indeed exists between two specific nodes, and thus, they need to access a low-level representation of the raw data and understand the mechanism of edge computation.

Our final design decision is to adopt a direct manipulation approach akin to that of Polaris,[21] Tableau,[22] and the visualization schemas approach.[38] Analysts directly interact with high-level conceptual representations of the relational data schema and indicate intention by manipulating these representations.

To create nodes, analysts drag and drop selected columns from the data management view to an empty area in the network schema view (Figure 2). 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 two 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)).

Analysts specify aggregation and projection, two transformative operations on existing networks, using



(a)

[✓ Aggregate two] AppointStart nodes if their difference < 3d

Aggregate AppointStart nodes by binning

Aggregate AppointStart nodes if they have the same attribute

(b)

Connect a LastName, FirstName with a LastName, FirstName

if they both connect to a AppointStart

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

dialog interaction rather than drag and drop. We make this design choice considering the fact that it is difficult to articulate these two operations within the network schema view. Dialog-based interfaces provide text-based controls that are easy to understand. Our design uses combo boxes to let analysts specify the type of nodes they want to perform these operations on.

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.

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 14). If the dimension used contains [many distinct] categorical values, the large number of subnetworks can lead to usability and performance problems. In our current design, users can scroll to see subnetworks hidden from the current viewport. Systems such as ManyNets[28] will be useful to visualize summary statistics of these subnetworks for easy comparison.

Visual encoding

The direct manipulation interface supports the articulation of operations that determine the constituent data of desired networks. Subsequently, it is important to visualize these networks appropriately. Commonly used visual variables to encode data dimensions are color, size, and spatial positions. In particular, spatial position encoding, or graph layout, plays an important role in showing prominent visual structures such as clusters and outliers.

The node-link representation included in Ploceus supports five layout algorithms: Fruchterman–Reingold force-directed layout,[42] circular layout, spring layout, the Kamada–Kawai algorithm,[43] and Meyer's self-organizing layout.[44] The effectiveness of these layout algorithms often depends on the specific properties of the network being visualized, and analysts can experiment with different algorithms through a combo box.

In addition to providing mechanisms for spatial position encoding, Ploceus intelligently infers the appropriate visual encodings for the nodes based on the type of the underlying table dimensions. Studies have examined how people perform in perceptual tasks in terms of accuracy when different information types (quantitative, ordinal, and nominal) are represented using different visual variables (e.g. area, color, and density).[45,46] Researchers have explored the issue of automatic graphic encoding[46,47] by incorporating these established design principles into system logic for effective visualization.

Ploceus draws from these research findings to apply effective graphic presentations in the visualization of networks. As noted in the previous section, when analysts create a new type of node, Ploceus automatically assigns a new color to that type. In addition, when analysts designate table column(s) as attributes of nodes, Ploceus analyzes the type of the column(s) to choose a visual mapping. Currently, Ploceus supports four types of column types: integer, float, date, and text (string). If analysts assign a quantitative column (integer or float) as node attribute, Ploceus will sum up the quantitative values and represent it using node size. For example, after adding an attribute Size to the people nodes constructed from LName, FName in Table 1, the node "Smith, John" will have a value of 381 for the Size attribute. Figure 6 shows the resulting visualization based on a larger dataset of the White House visitor information. When analysts designate a date column as a node attribute, Ploceus represents dates using node size by converting date values into the number of milliseconds since 1 January 1970, 00:00:00 GMT.

Encoding a categorical node attribute is a design decision requiring more consideration. It is arguably best practice to use visually distinct colors to represent a categorical variable.[48] Alternative ways of encoding categories are to use texture or shape.[46] In any case, when there are many unique categorical values, it is difficult to define enough visually distinct representations.

In Ploceus, we use color to represent the type of node as discussed in section "Design of direct manipulation interface." This initial decision implies that we may have to use shapes to represent categorical node attribute values. The number of visually distinct shapes, however, is limited compared to the available choice of distinct colors.[49] Assuming that analysts usually will create relatively few node types, we experimented with using shape to encode node type and using color to encode the categorical attribute values instead. Figures 7 and 8 show visualizations generated with this approach. Figure 7 shows a network of the White House visitors (represented as circles) and visitees (represented by diamonds). Ploceus assigns a default color to all the nodes. Figure 8 shows the resulting network after we add an attribute denoting the meeting location to the visitor nodes, which is represented using color.

However, informal feedback gathered from visualization experts on this approach was not positive. They strongly preferred encoding node type as color instead

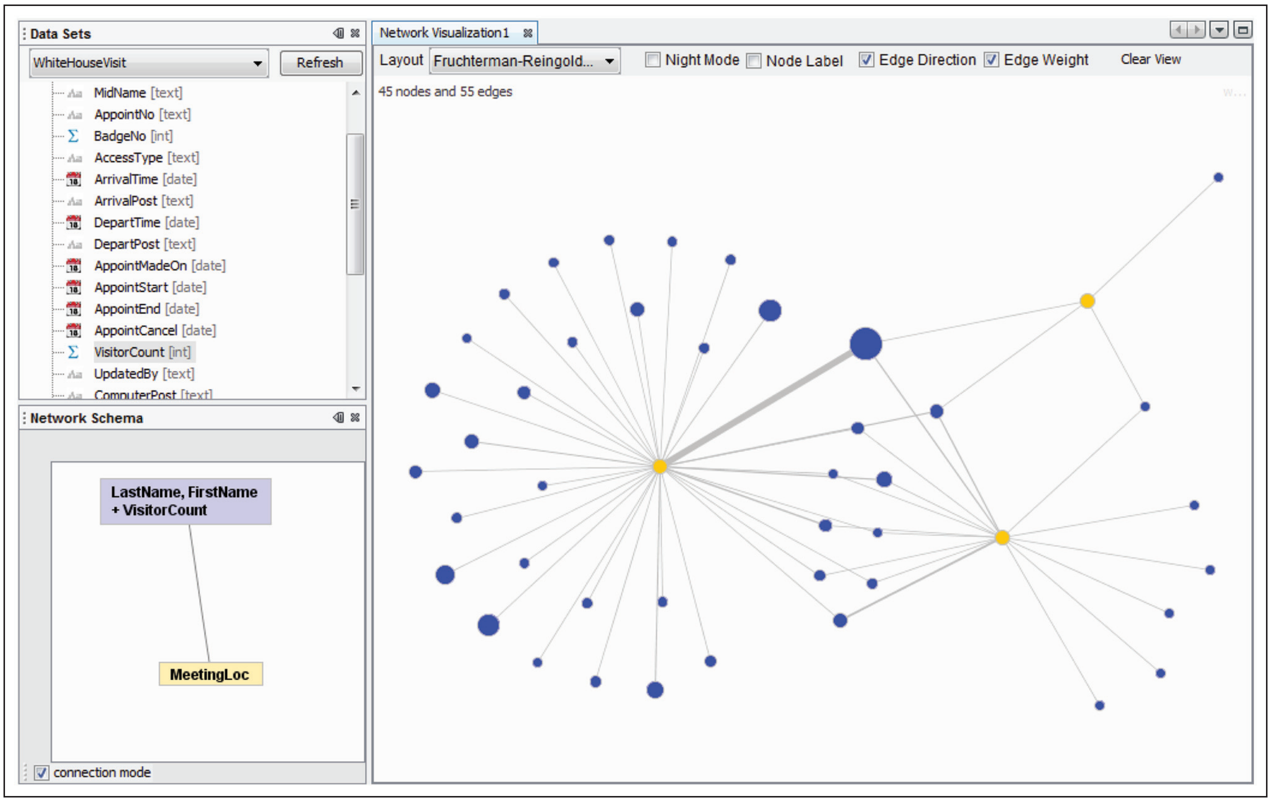


Figure 6. Ploceus visualizes the attribute VisitorCount of the visitor nodes constructed from LastName, FirstName as node radius (size).

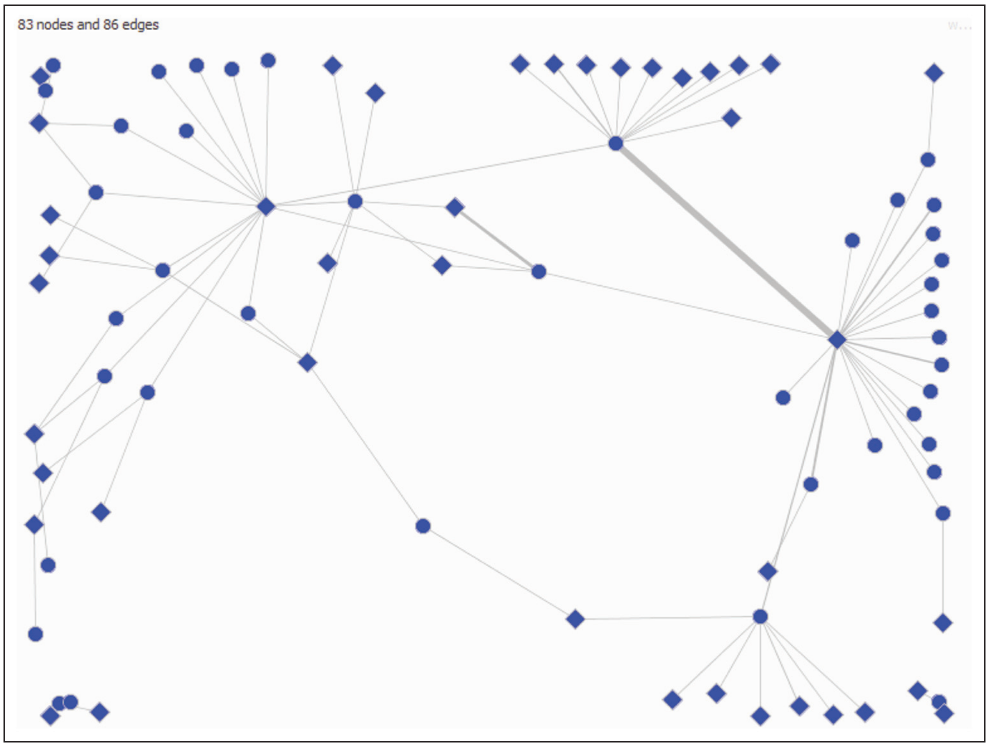


Figure 7. A network of the White House visitors, represented as circles, and visitees, represented by diamonds.

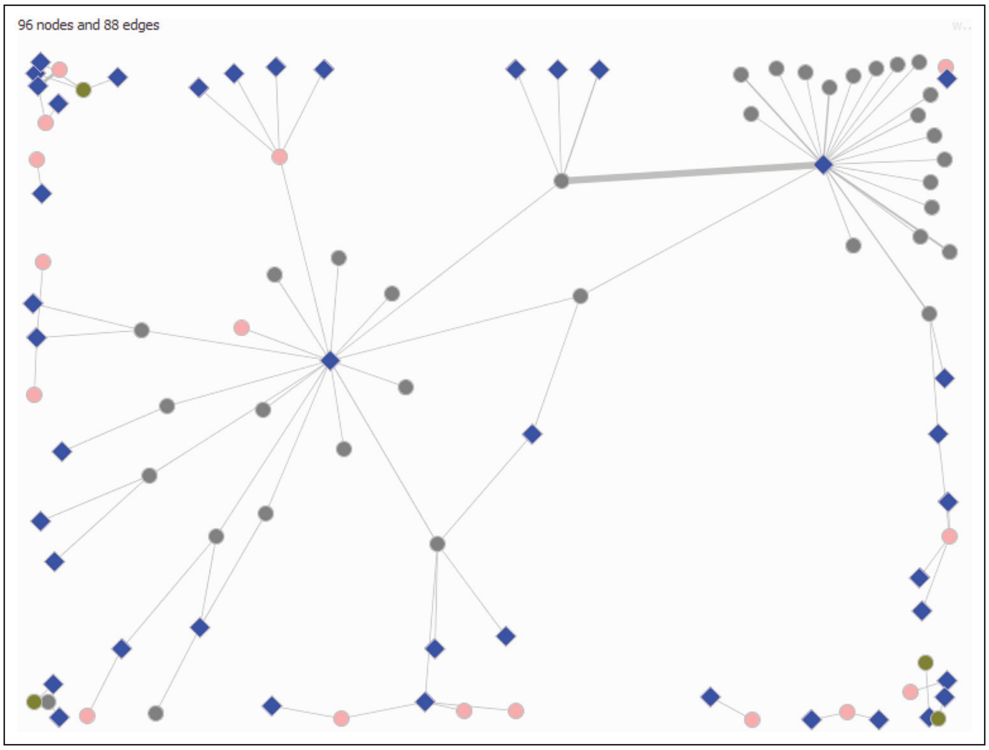


Figure 8. Adding "meeting location" as an attribute to the circular visitor nodes and representing the attribute using color.
Visitee nodes remain as blue diamonds.

of shape and suggested that it was potentially confusing to interpret (Figure 8). Since node type can also be considered as a node attribute and is automatically generated, it is more essential than an optional node attribute defined by users. We thus decided to treat node type as the default node attribute and to continue encoding it using color. Analysts can define new attributes by dragging and dropping columns onto the existing nodes, and if the new attribute is categorical, it will be color coded and replace the default color assigned to the node type. Users can change which node attribute to encode through a pop-up menu. Figure 9 shows the visitor–visitee network, where visitees are in yellow and the visitor nodes are colored by the locations of their visits. In our current implementation, Ploceus only supports one active attribute for a node type: adding a new attribute will replace any existing attribute assigned to the node type. This design decision was made to keep the current implementation tractable. In future versions of Ploceus, we plan to remove this constraint.

The study described in this section lays the groundwork for further investigation of a comprehensive graph visualization framework. The Polaris formalism[21] establishes an algebraic framework for table-based visualizations that provides effective mapping from data variables to visual variables. We envision that a similar framework is possible and is needed to describe the mappings between attribute relationship graphs and various graph visualizations. Such a framework will be useful for automated generation of graph visualizations and may suggest visualization techniques that have not been explored before.

Edge semantics and construction strategies

With such a set of diverse operations provided, 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 straightforward. For example, 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)

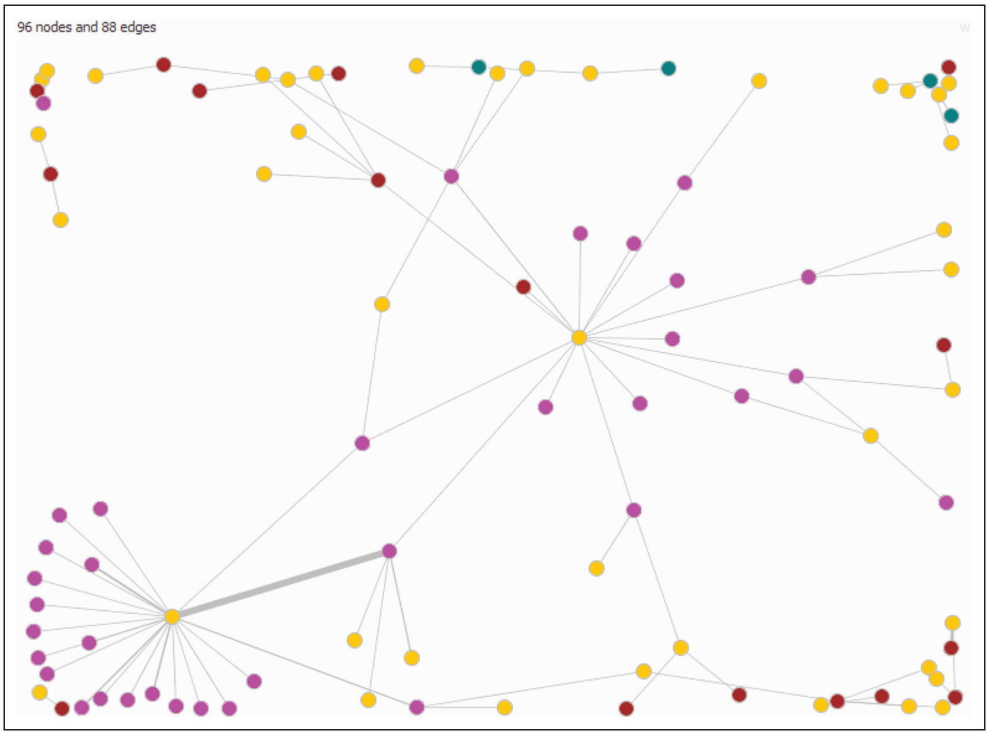


Figure 9. Ploceus visualizes the attribute MettingLoc of the visitor nodes constructed from LName, FName as node color. The visitee nodes are in yellow, and the visitor nodes are colored by the locations of their visits.

already defines an explicit network relationship of Researcher×Grant (or GID×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×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, and the edge weight is determined by the number of occurrences of the (GID, PID) pair.

However, this weight may not be at the right level of abstraction to the analyst, as Sharad Mehrotra and Padhraic Smyth have collaborated 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 and 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 10(a) shows the label for the first case and Figure 10(b) shows the label for the second case discussed in this section.

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.

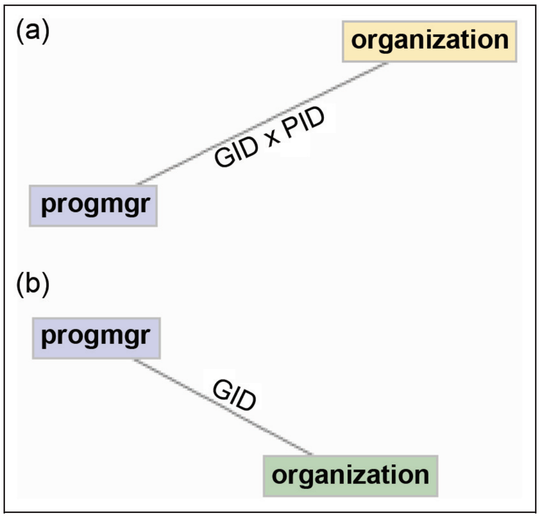


Figure 10. Edge semantics labels in the network schema view: (a) the edge weight between a program manager *p* and an organization *o* denotes the number of researchers from the organization *o* who have received grants awarded by *p* and (b) the edge weight represents the number of unique grants that a program manager has given to an organization.

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. Every network generated can be saved as a GraphML file and be reloaded into Ploceus. Figure 11 shows an overview of the work flow in using Ploceus.

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 IIS division from 2000 to 2003. A subset of the data is presented in Table 3. It is

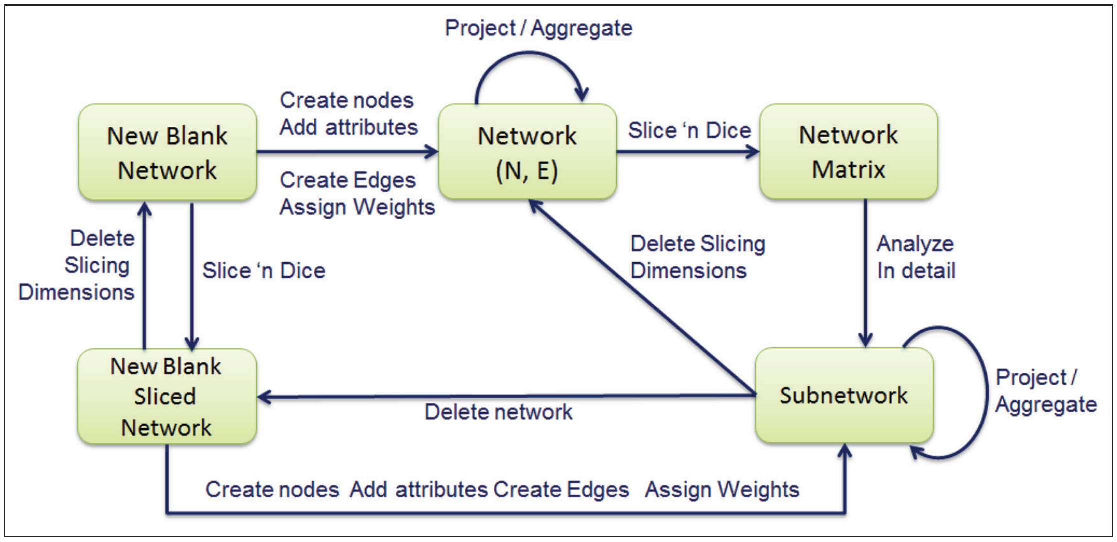


Figure 11. An overview of the work flow in using Ploceus.
Different states of the network are shown in rectangles, and the arrows represent the user interaction to transit between the states.

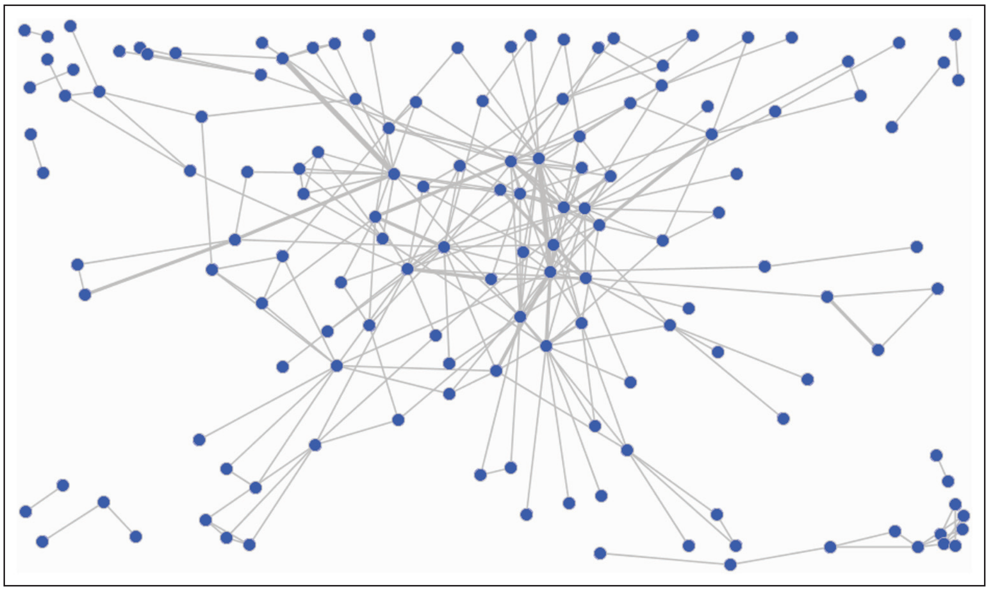


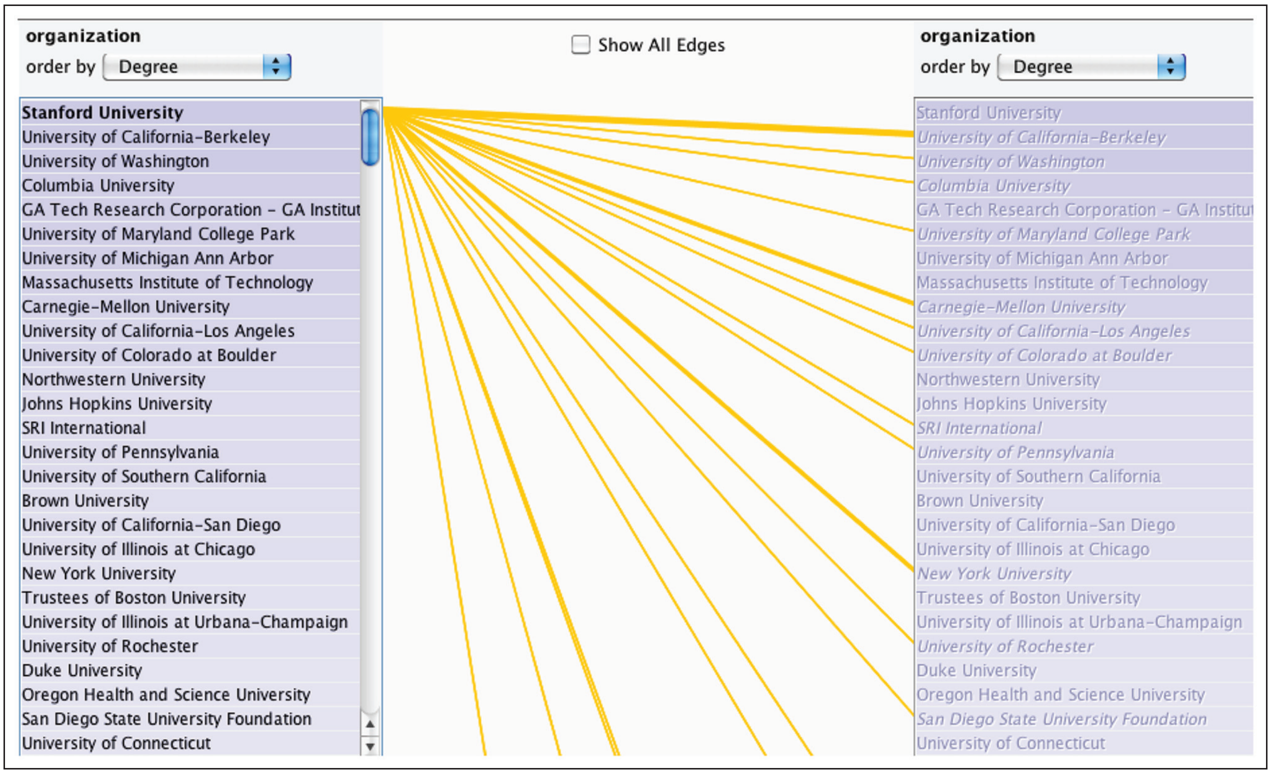
Figure 12. Collaboration between organizations on NSF IIS grants, 2000–2003.
NSF: National Science Foundation; IIS: Information & Intelligent Systems.

a long-standing policy of NSF to encourage interinstitution 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 dataset 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 12 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 (Figure 13), we see that Stanford University, University of California Berkeley, University of Washington, Columbia University, and Georgia Tech are the top five cross-institution collaborators. 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.

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 14. 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 4 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 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.

To investigate further, we right click in the 2003- large grant cell and choose "Analyze in detail" to


organization
order by Degree

Show All Edges

organization
order by Degree

Stanford University
University of California-Berkeley
University of Washington
Columbia University
GA Tech Research Corporation - GA Institut
University of Maryland College Park
University of Michigan Ann Arbor
Massachusetts Institute of Technology
Carnegie-Mellon University
University of California-Los Angeles
University of Colorado at Boulder
Northwestern University
Johns Hopkins University
SRI International
University of Pennsylvania
University of Southern California
Brown University
University of California-San Diego
University of Illinois at Chicago
New York University
Trustees of Boston University
University of Illinois at Urbana-Champaign
University of Rochester
Duke University
Oregon Health and Science University
San Diego State University Foundation
University of Connecticut

Stanford University
University of California-Berkeley
University of Washington
Columbia University
GA Tech Research Corporation - GA Institut
University of Maryland College Park
University of Michigan Ann Arbor
Massachusetts Institute of Technology
Carnegie-Mellon University
University of California-Los Angeles
University of Colorado at Boulder
Northwestern University
Johns Hopkins University
SRI International
University of Pennsylvania
University of Southern California
Brown University
University of California-San Diego
University of Illinois at Chicago
New York University
Trustees of Boston University
University of Illinois at Urbana-Champaign
University of Rochester
Duke University
Oregon Health and Science University
San Diego State University Foundation
University of Connecticut

Figure 13. Collaboration between organizations on NSF IIS grants, 2000–2003, in a list representation.
NSF: National Science Foundation; IIS: Information & Intelligent Systems.

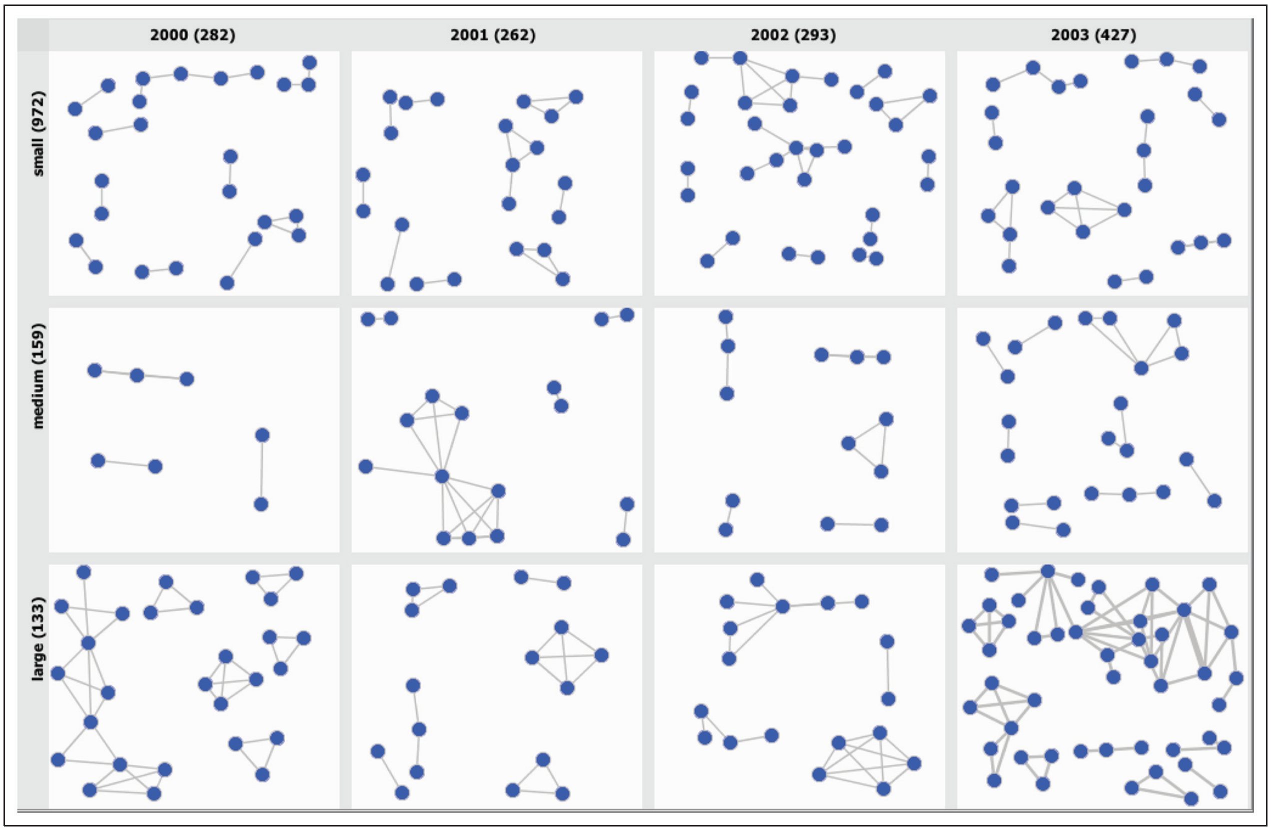


Figure 14. Collaboration between organizations on NSF IIS grants, broken down by year and amount.
NSF: National Science Foundation; IIS: Information & Intelligent Systems.

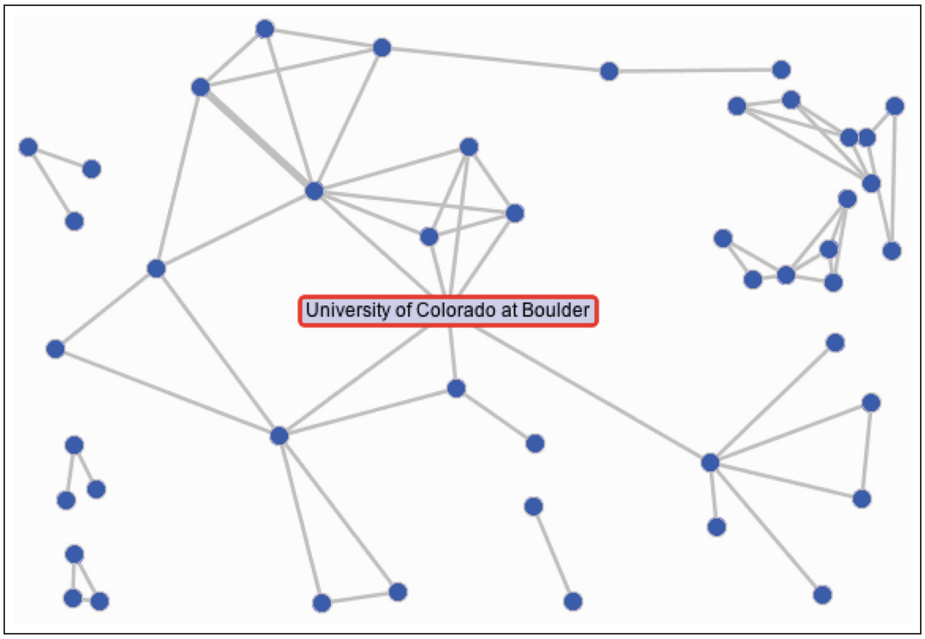


Figure 15. CU Boulder is an important actor in the 2003–large grant collaboration network
CU: University of Colorado.

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 15). 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 16).

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 grants is small grants. After filtering out noncollaborating institutions, we find that grants awarded by them do not particularly show greater activities of collaboration (Figure 17). It is also obvious from the visualization that Ephraim Glinert has awarded a number of grants to groups of four institutions (visualized in the form of tetrahedra), and Stephen Griffin awarded one grant to a group of five collaborating institutions (in the form of a pentahedron). Such patterns, some of which are highlighted in Figure 17, are not seen in grants awarded by other program managers (including those hidden from the current view and have to be revealed by scrolling).

Extending to one-mode networks

One-mode networks as reflexive relational tables

The discussion and scenario so far focus on modeling and visualizing multimodal networks from tabular data. In these tabular datasets (see Tables 1–3), the relationship types among the entity types are *binary*, that is, the relationships are defined between two different classes of entities. Using the projection operator provided in Ploceus, we can create one-mode networks from multimodal networks. The system, however, did not initially provide direct support for modeling networks from tabular data that contain a *unary* relationship, defining references within one class of entities. In the ER data model, such a relationship is called a reflexive or recursive relationship.[2]

Table 5 shows a sample reflexive relational dataset of an egocentric social network of the user “jsmith” on Twitter. Table 5(a) records information about each

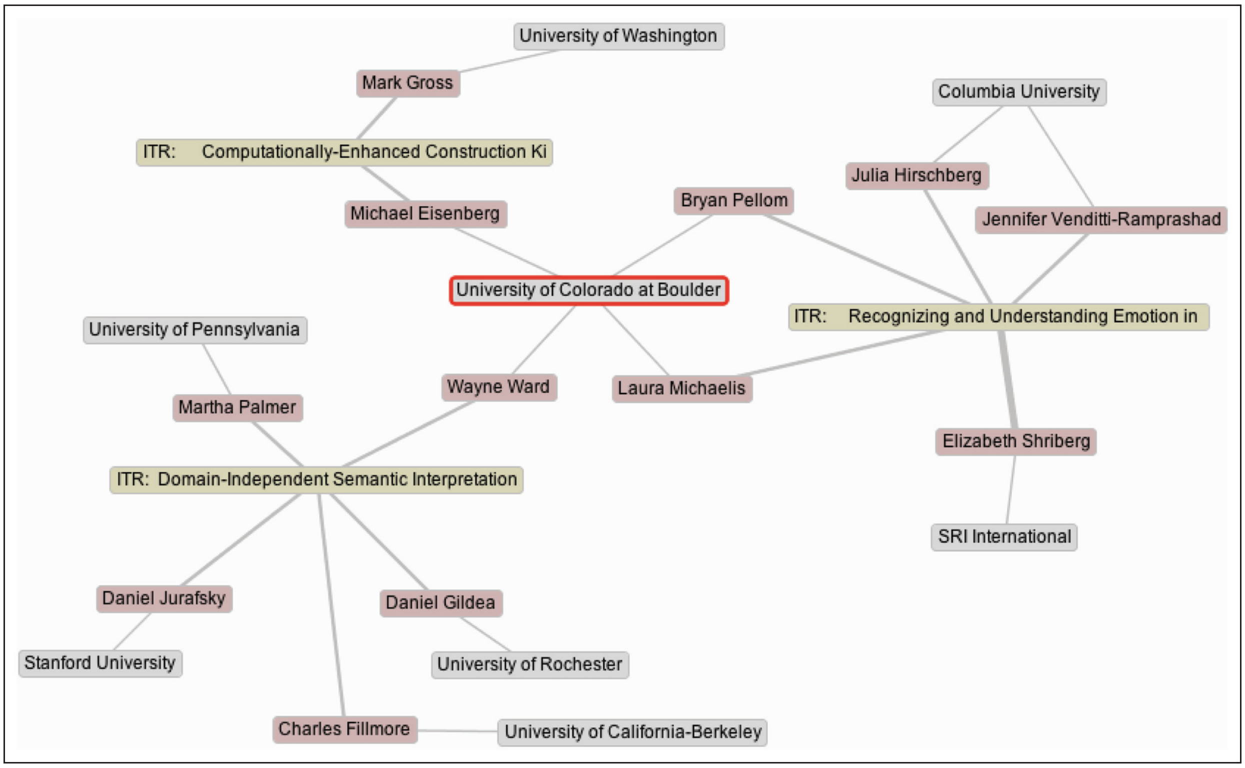

University of Washington
Mark Gross
ITR: Computationally-Enhanced Construction Ki
Michael Eisenberg
Bryan Pellom
Columbia University
Julia Hirschberg
Jennifer Venditti-Ramprashad
University of Colorado at Boulder
ITR: Recognizing and Understanding Emotion in
University of Pennsylvania
Wayne Ward
Laura Michaelis
Martha Palmer
Elizabeth Shriberg
ITR: Domain-Independent Semantic Interpretation
SRI International
Daniel Jurafsky
Daniel Gildea
Stanford University
University of Rochester
Charles Fillmore
University of California-Berkeley

Figure 16. Large grants received by CU Boulder and other institutions in conjunction in 2003.

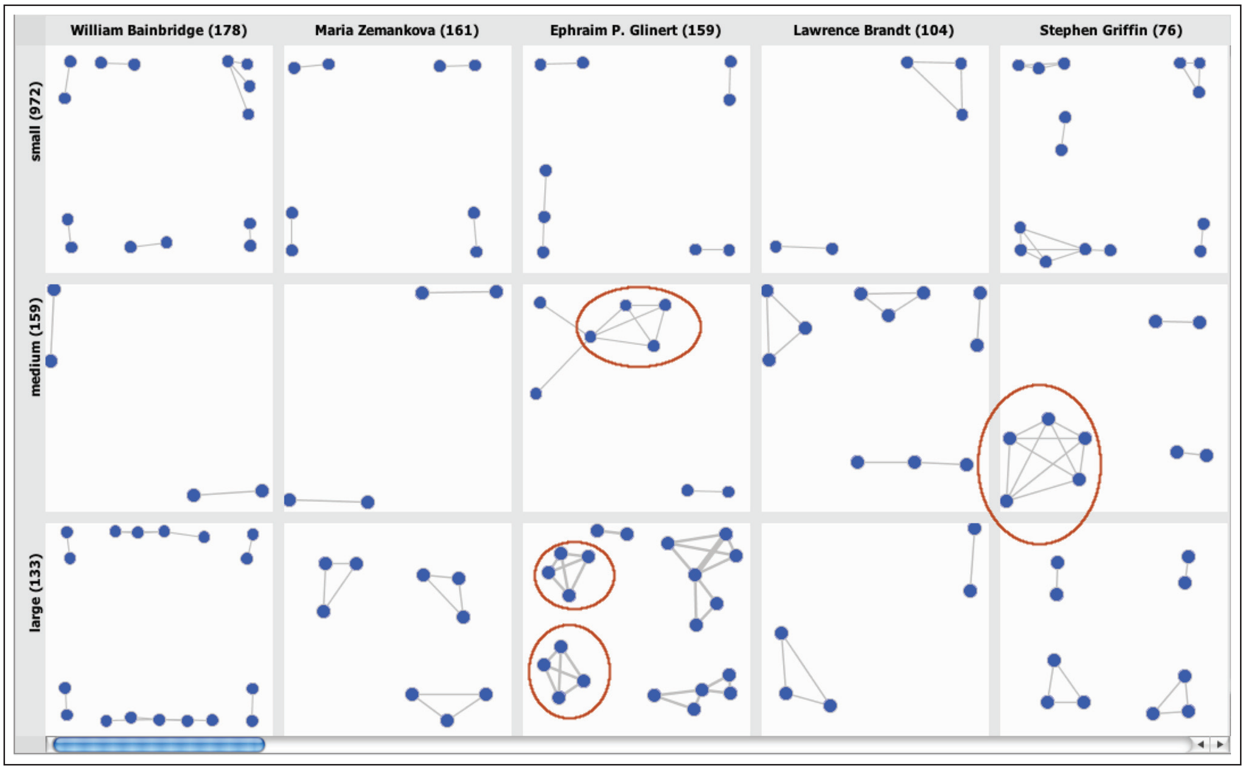


Figure 17. Collaboration between organizations on NSF IIS grants, broken down by program manager and amount. Cliques with more than three nodes are highlighted.

Table 5. Two tables describing relationships between individuals on Twitter.

(a) Person

ID	Join_Date	Favorites	Tweets	Location
jsmith	22 Feb 08	2	24	Silicon Valley
fwong	4 Apr 08	20	231	West Lafayette
jwallace	18 Nov 09	6	120	Finland
ajabbar	25 Jun 10	30	15	Paris, France
suzuki	28 May 09	9	567	San Francisco, mostly

(b) Relationship

Source	*Target*	*Relationship*	*Relationship_date*
jsmith	ajabbar	Following	11 Jan 11
fwong	jsmith	Followed	16 Jul 11
jwallace	jsmith	Mention	1 Nov 11
ajabbar	jsmith	Followed	2 Sep 10
jsmith	fwong	Mention	5 Feb 10

Twitter user including the account (ID), the date when the user joined Twitter (Join_Date), the number of tweets designated as favorites by the user (Favorites), the number of tweets by the user (Tweets), and the self-described location (Location). Table 5(b) records the relationships between the Twitter users. We can consider this dataset to be a *one-mode directed* network. Such data are pervasive given the proliferation of social network sites and social media, and Ploceus should provide reasonable means to incorporate these datasets.

Design considerations for modeling and visualization

Prior study such as PivotGraph[29] enables analysts to perform attribute-based node aggregation in a one-mode network through combo boxes. Since Ploceus provides more operations with greater flexibility, simple user interface controls may not suffice. We thus focus on extending the current interface design to support one-mode networks.

Incorporating one-mode networks into Ploceus' interface and interaction framework turns out to be relatively straightforward. Following the convention of representing one-mode networks as separate node and edge tables,[50] the data management view of Ploceus shows individual columns of the node and edge tables of a one-mode network, respectively (Figure 18). To provide a consistent experience in modeling both multimodal and one-mode networks, we continue utilizing the direct manipulation paradigm. To construct a Twitter network, for example, analysts follow similar steps to those outlined in section "Design of direct manipulation interface." They first drag and drop the ID column to the network schema view, and this operation adds all the Twitter users in the dataset to the network. In order to create connections, analysts must drag and drop the ID column again to the network schema view to create a dummy node. Ploceus recognizes that these two ID nodes in the schema view come from the same table column, thus treating them as the same type and assigning the same color. Finally, analysts click on one of the ID nodes in the schema view and then drag and release the mouse button on the other ID node to create edges. Since this is a directed network, Ploceus supports creating directed edges when analysts hold down the "ctrl" key while using the mouse to connect nodes. Ploceus infers the condition to join the node and edge tables and creates connections accordingly.

Potentially, there is an alternate way to model the same network. Instead of adding ID twice to the network schema view, analysts can drag and drop the Source and Target columns to the schema view, respectively, and connect these two columns. Since Source and Target are distinct columns, Ploceus will treat the nodes created from these two columns as having different types, which is counterintuitive. Furthermore, the source and target columns in edge tables are often numerical identifiers that refer to individual entities in node tables. Node labels created from these columns therefore are not intelligible to analysts. Based on these considerations, we decide not to pursue this approach.

Considering these factors, we make the following two design decisions. First, if analysts provide the one-mode data by importing files formatted for graph data (e.g. GraphML[51] and Pajek[10]), Ploceus parses the data and populates the node and edge tables. During this importing process, Ploceus marks the Source and Target columns in the edge table as hidden, and these two columns are absent in the data management view to prevent analysts from this kind of modeling strategy. Second, if analysts provide the one-mode data by

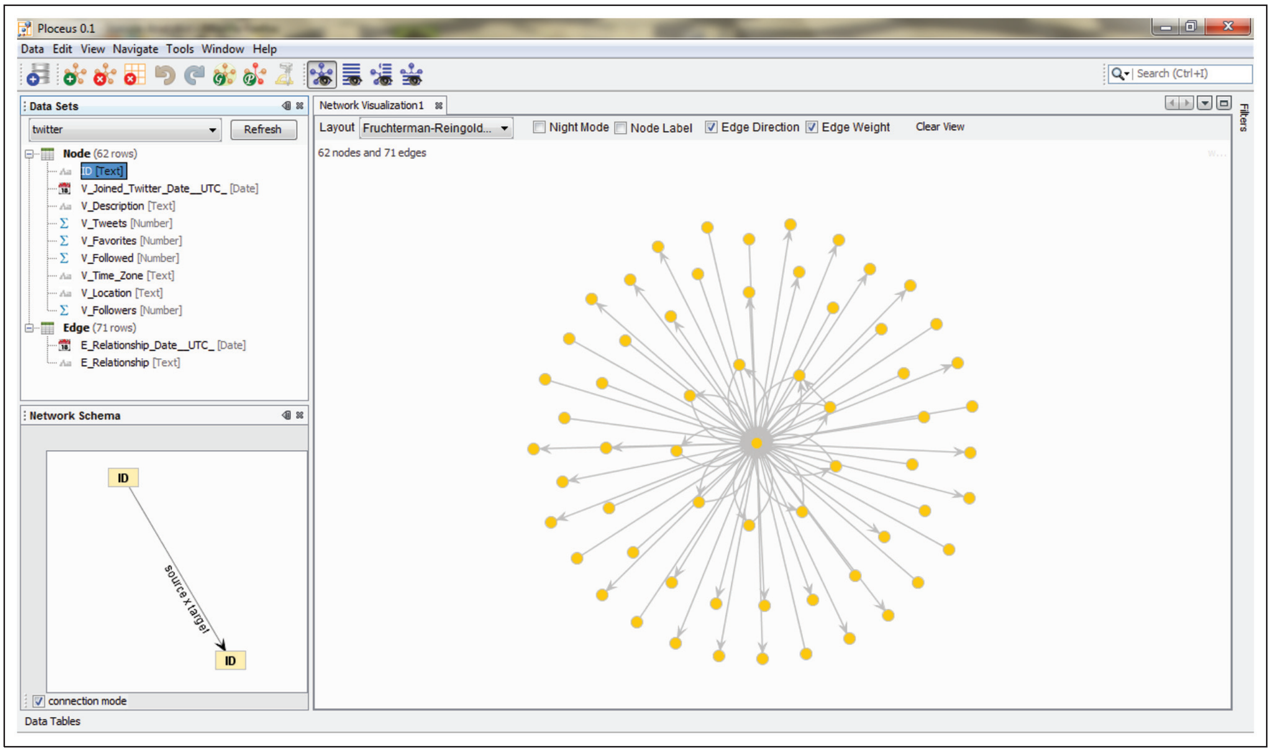


Figure 18. Using Ploceus to construct a one-mode directed Twitter network.
The arrows indicate the directions of "following" on Twitter.

pointing Ploceus to an existing relational database comprising multiple tables, it is a much more difficult inferencing problem to identify the Source and Target columns. In this case, Ploceus allows analysts to aggregate two or more different types of nodes under a self-defined node type. Similar to the default aggregation discussed in section "Operations," this "aggregate type" operation merges nodes if they share identical labels and attribute values even when these nodes are of different types.

Figure 18 shows a resulting visualization of one of the authors' own Twitter network. Ploceus represents the direction of the edges using arrows. If two nodes are connected to each other in both directions, the two edges will overlap and potentially cause confusion. Ploceus thus renders these kinds of edges as quadratic Bézier parametric curves, so that bidirectional edges do not overlap and form a distinctive visual pattern (Figure 18).

All the network operations such as adding attributes and slicing 'n dicing still also apply for one-mode directed networks. Figure 19 shows three egocentric subnetworks generated by slicing 'n dicing the network in Figure 18 using the (Relationship) dimension in the edge table: a "follower" network, a "following" network, and a "mention" network. In addition, the people nodes have an attribute "Tweets," representing the number of tweets by each user, encoded as node size.

Computing connections

Ploceus is powered by the implementation of a formal framework that systematically specifies how to compute edge connections and assign edge weights. In this section, we provide an overview of the framework for readers interested in the implementation details.

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. both 142 and 16 are 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 OEOB. When Data Mining of Digital Behavior and

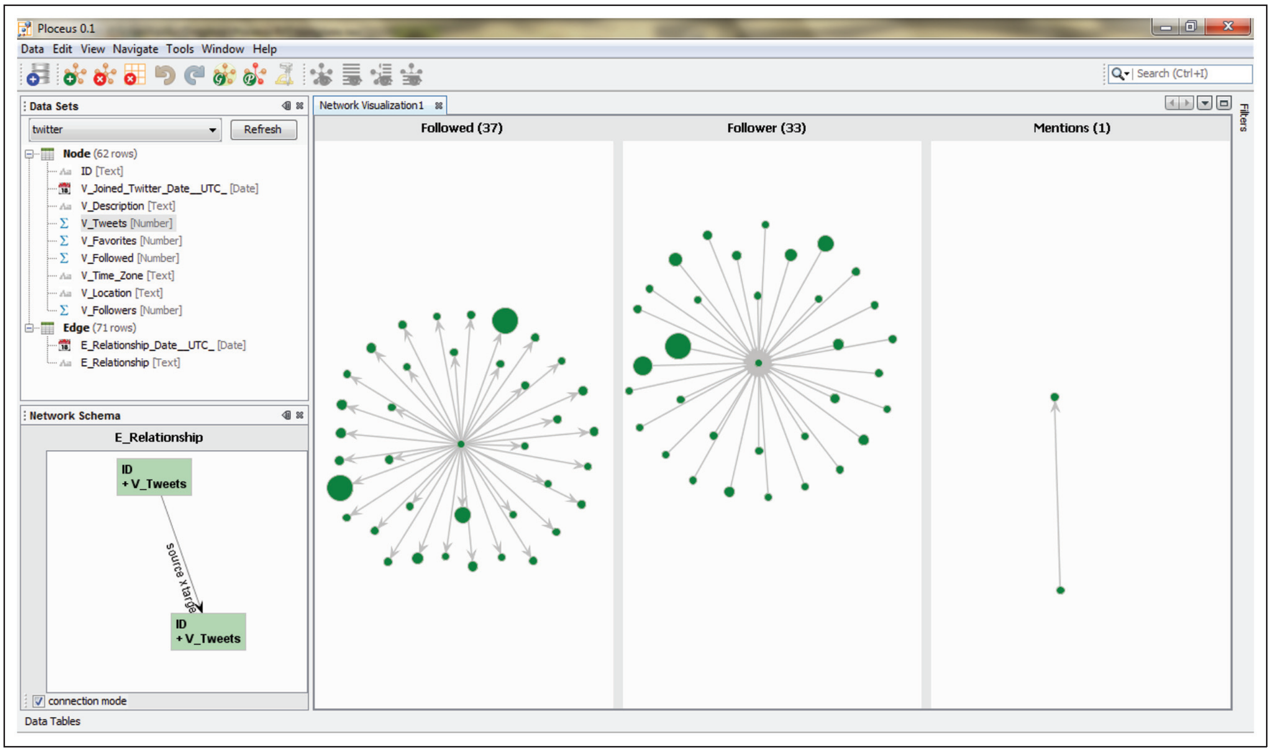


Figure 19. Slicing ′n dicing the twitter network using the (Relationship) dimension and encoding node size as the number of tweets.

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 titled "Data Mining of Digital Behavior" 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 also co-occurs with A.

We base our formal framework on the relational model[52] used widely in database theories, with basic relational algebraic operators such as selection (σ), projection (π), join ($\bowtie$), and aggregation ($\mathscr{F}$).[2] We make the following three assumptions:

- Each row in a table has a unique identifier;
- Each value in the table cells is *atomic*, that is, 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.

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 returning "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, which is 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

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

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

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

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[2] 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.

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 "Operations," 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 that the analysts have specified a function of aggregating nodes, the newly produced nodes will inherit the locales of the nodes being aggregated

$$\mathit{locale}(n') = \mathit{locale}(n_1) \cup \ldots \cup \mathit{locale}(n_j)$$

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

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

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

$$\exists m \in M, e(n_1, m) \in E \text{ and } e(n_2, m) \in E \Rightarrow e(n_1, n_2)$$

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

$$\mathit{basis}(n_1, n_2) = \{m \in M | e(n_1, m) \in E \text{ and } e(n_2, m) \in E\}$$

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.

Extending to one-mode graphs

We initially developed the formal framework without giving serious consideration to the possibility of extending to one-mode graphs. Scenarios discussed in section "Extending to one-mode networks" make a compelling case to broaden the scope of our framework for the construction of directed graphs from data tables describing reflexive relationships. Section "Extending to one-mode networks" presents our design rationale at the interface level; in this section, we briefly discuss the underlying theoretical logic.

Suppose we want to construct a graph showing the Twitter relationships between different users from Table 5. We first create two identical sets of first-order Person nodes from Table 5(a), called N and N_1. To construct connections between these two sets of first-order nodes, we follow the definition discussed in section "First-order graphs"

$$\text{A node } n \in N \text{ is connected to a node } n_1 \in N_1$$
$$\text{if } locale(n) + locale(n_1) \in R_P \bowtie R_P \text{ on the condition that}$$
$$R_P \cdot \mathrm{ID} = R_E \cdot \mathrm{Source} \quad \text{and} \quad R_P \cdot \mathrm{ID} = R_E \cdot \mathrm{Target}$$

where R_P represents the person table and R_E represents the relationship table

Note that here we make a slight modification by replacing the union operator $\cup$ with the concatenation operator $+$ in $locale(n) + locale(n_1)$. Concatenation preserves the order of the values in the tuples and does not remove duplicates, thus ensuring that the order of tuple values is preserved in edge creation. We can then assign directions to edges by specifying n nodes as source nodes and n_1 nodes as target nodes. Finally, we can do an aggregation of n and n_1 nodes if they share the same label.

Implementation

Ploceus is built entirely in Java on the NetBeans Rich Client Platform.[53] It utilizes two major external toolkits and libraries: H2[54] as the underlying database for relational algebraic queries and JUNG[25] 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 Structured Query Language (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 datasets. 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 study includes optimization of the implementation of operations.

Outstanding problems and limitations

Joining multiple tables

When computing connections between columns from different tables, we currently infer equi-join conditions by analyzing foreign key constraints between tables through a Dijkstra shortest-path algorithm.[39] We first construct a graph where the nodes are the columns in each table, and primary key columns and foreign key columns are connected. Given two columns to be connected, we then apply the shortest-path algorithm on this graph.

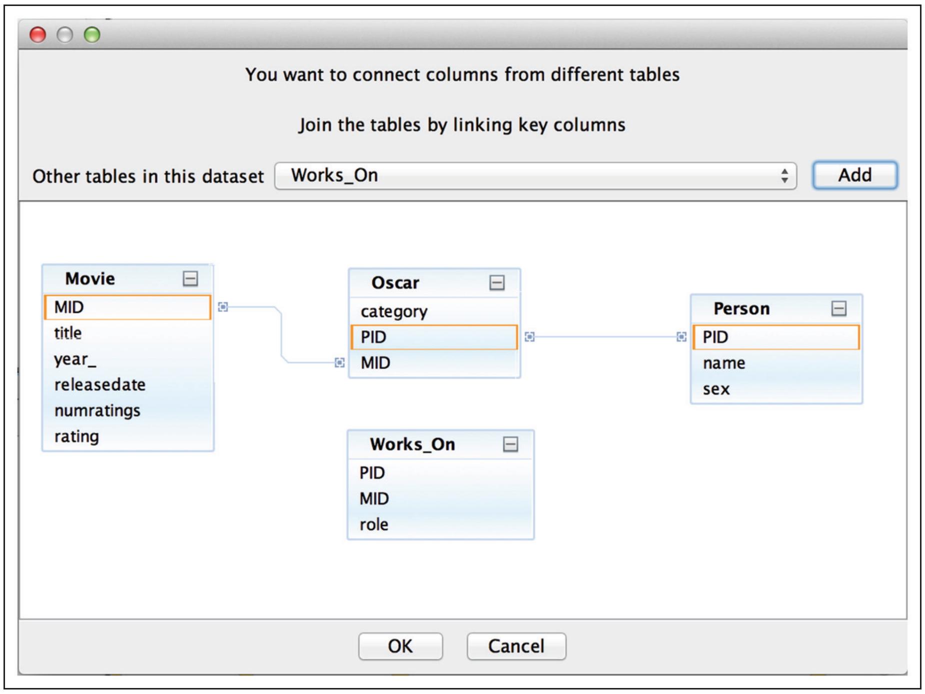
When a database becomes more complex in terms of ER modeling, there might be multiple reasonable equi-join conditions. It is thus the user's decision to choose the appropriate join condition. Currently, Ploceus handles this situation through a dialog, letting users interactively add relevant tables and connect the primary keys and foreign keys to specify the desired join condition (Figure 20). Related systems such as Tableau[22] take a similar approach. Tableau does not support interactive table joining during the process of exploration. Instead, at the stage of data import, analysts need to explicitly join tables to include all the data columns necessary for exploration. Tableau supports a richer set of join types and conditions (Figure 21).

Potentially, we can also use more sophisticated techniques to automatically compute a number of different join conditions and to rank them by inferring analysts' intention. The diversity of all the possible join conditions, however, can hardly be fully captured. More importantly, all these approaches do not address a fundamental issue satisfactorily; analysts must have a precise and good understanding of the concepts of relational join, primary key, and foreign key. Even if they understand these concepts well, it is still nontrivial to interpret the semantics of edges constructed as a result of joining multiple tables.

Currently, we do not have a satisfactory solution to this problem, and we doubt there will be one if the underlying data model is going to be relational. The recent emerging NoSQL databases[55] might provide an interesting angle to address this issue. Multiple related tables in relational databases are a direct consequence of the design choice to normalize data tables for the sake of minimizing redundant data representation and avoiding anomalies in data modification.[2] These goals are not emphasized in NoSQL databases. It would be interesting to explore if the problem of mandatory join specification can then be eliminated if we choose NoSQL data model (key-value pairs instead of data tables) where the joins have been done a priori, in some sense eliminating to do them explicitly.

Big/dense graphs

Scalability is an important issue in Information Visualization and Visual Analytics research. An implication of the network modeling power provided by Ploceus is that analysts can easily create both big graphs with thousands or even millions of nodes and dense graphs where the number of edges is close to the maximum possible number of edges. Visualizing and analyzing these graphs remain great research challenges. Due to the limited number of pixels available on computer screens, it is impossible to display all the nodes without resulting in cluttering or overlapping.

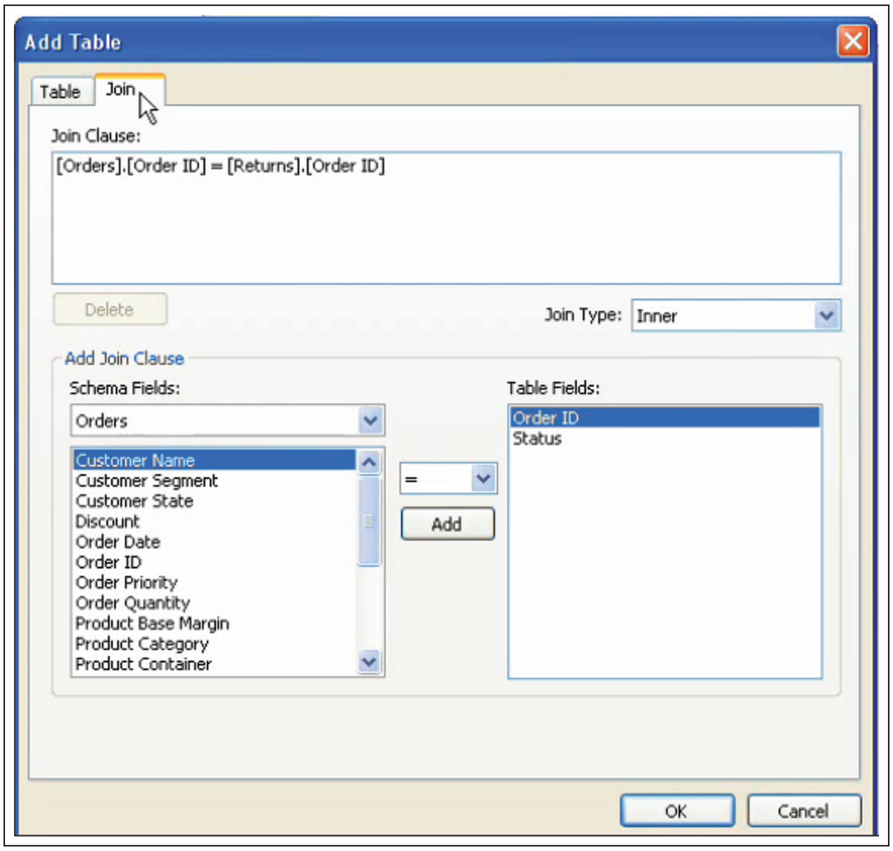


You want to connect columns from different tables

Join the tables by linking key columns

Other tables in this dataset: Works_On

Figure 20. Users interactively add relevant tables and connect the primary keys and foreign keys to specify the desired join condition in Ploceus. In this example dataset about IMDB movies, a person (PID) and a movie (MID) can be connected in two different ways, via the Works_On table and the Oscar table, respectively.
IMDB: Internet Movie Database.



Add Table

Join Clause:

[Orders].[Order ID] = [Returns].[Order ID]

Join Type: Inner

Add Join Clause

Schema Fields: Orders

Table Fields:

Figure 21. Dialog interface in Tableau to join multiple tables.

Dense edges also cause severe performance issues in computing graph layout. In Ploceus, the problem of scale is handled using subnetwork sampling. The interface displays bold messages to remind analysts that only a subnetwork is shown. Analysts can interactively add or remove subnetworks through search queries.

A few potential directions exist for future research on this problem. First, it is worthwhile to investigate appropriate mechanisms of network sampling. Ploceus currently samples randomly. Techniques such as the degree-of-interest functional retrieval[56] sound more promising. Users can pick a focal point and the system displays a subgraph that is of maximal interest. Second, we would like to investigate when it is actually useful to show an overview of the entire network and to articulate the user tasks involved in these situations. It may be possible that we can design alternative visual representations that provide information needed in these tasks without having to show every single node and edge. Third, one way to analyze big graphs is to use a divide and conquer strategy by breaking the graphs down into meaningful subgraphs. Filtering and slicing 'n dicing are two reasonable mechanisms to do so, and they are included in Ploceus. It may be necessary to analyze and compare multiple networks at the same time. Ploceus now organizes multiple networks in the form of a matrix, but there are potential readability and usability problems when each of these networks contains a large number of nodes and edges. While systems such as ManyNets[28] have taken a first step in the effort of facilitating visual analysis of multiple networks, more research is needed to understand and design for multiple network analysis.

Expressive power

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 5 years, the set of operations described in section "Operations" is not sufficient to express such semantics of conditional connectivity.

Further study 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.[2] 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.

User evaluation

To understand the implications of the algebraic framework and the interface design, we conducted an evaluation of the learnability and usability of Ploceus. We recruited 10 participants, including one undergraduate student; five graduate students in the areas of computer science, communication design, and ergonomics; and four working professionals in the areas of software engineering, program managing, and electrical engineering. Eight of them were knowledgeable of database technologies and SQL queries, and the other two did not have relevant experience in these technologies. All of them had never seen or used Ploceus. The goal of evaluation was to identify qualitative insights about the way people think about network visualization construction and how well Ploceus supports network modeling.

Tasks and procedures

We gave a brief introduction to Ploceus, demonstrated the main functionalities of the system and showed the participants how to construct different networks using the White House Visitor dataset (Table 1 shows sample rows). We then asked them to create visualizations and answer the following questions on the NSF dataset (Table 3 shows sample rows):

- Can you create a visualization showing the collaboration pattern between organizations on research grants?
- Which organization(s) tend to collaborate the most?
- In which year(s) do we see the most cross-organization collaboration?

We explicitly told the participants that while there were three tables in the NSF dataset, they could connect any two columns in the data management view (Figure 2) as if these columns were from the same table; Ploceus could infer the right join condition in such simple datasets.

To answer the given questions, the participants needed to create visualizations in multiple steps as outlined in the scenario in section "Scenario: analyzing cross-institution research efforts": first, add the grant IDs and the organizations as nodes; then connect these two types of nodes; perform a projection on the grant IDs so that the organizations are connected directly to each other via common grant IDs, and finally slice and dice the network using the date dimension to break down the network by year.

We asked the participants to think aloud, observed their interaction with the system, recorded their interaction history as hand-written notes, and sought their impressions and comments on the system after they completed the tasks. Our aim was to gain an understanding of how difficult the system was to learn and use, if there were any problematic design issues, and how we might be able to address the difficulties experienced by the participants.

Results and analysis

Out of the 10 participants, four did not experience any difficulty and quickly answered the three questions accurately; five participants did experiment a few times before completing the tasks successfully. Only one participant failed to discover the correct strategy within half an hour.

In addition to task performance, we were interested in understanding the participants' strategies and thinking in action in visualization construction and exploration processes. Inspired by Norman's seven stages of action,[57] we hypothesized a model of expected action sequences in the process of visualization construction and exploration (Figure 22(a)). We anticipated that using Ploceus to construct network visualizations would involve the following steps:

1. *Interpret task/form intention.* When a task is assigned, users need to be able to understand what kind of insight is being inquired; when the task is self-initiated, users form an intention to look at certain aspects of the data.
2. *Simulate appropriate visualizations.* Users mentally simulate and visualize possible representations that can provide the desired insight.
3. *Identify relevant data dimensions.* Only a few selected columns in the dataset are relevant for a given question, users need to identify which data dimensions to include in the visualization.
4. *Conceive construction strategy.* Given a desired visualization, choose appropriate operations in the Ploceus framework and understand how these operations should be combined to create the visualization.
5. *Execute construction sequence.* Users execute the conceived construction strategy and perform each operation using the Ploceus interface.
6. *Interpret visualizations.* Finally, assuming that the construction is successful, users must be able to accurately interpret the visualizations generated and provide answers to the given questions.

We used this model as a framework to analyze and interpret participants' interaction histories with Ploceus, while keeping in mind that the model needed to be validated and refined based on empirical evidence. The model was useful for deductive analysis and helped us pinpoint difficulties the participants experienced at various stages of action; on the other hand, we also identified evidence pointing to possible refinements of the model.

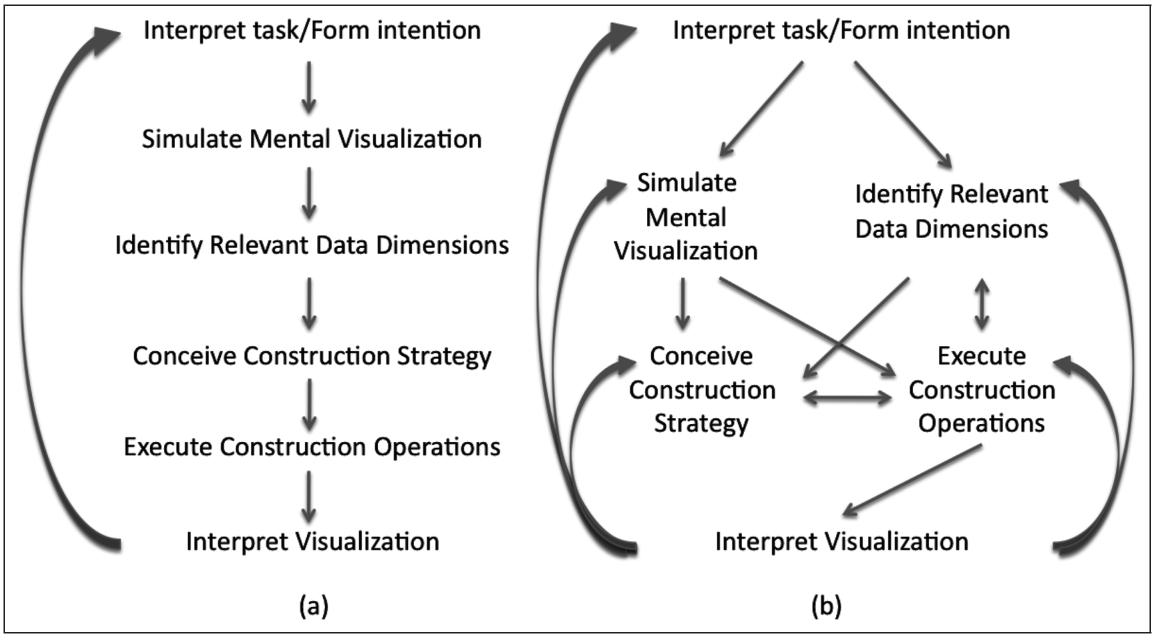

Interpret task/Form intention
Simulate Mental Visualization
Identify Relevant Data Dimensions
Conceive Construction Strategy
Execute Construction Operations
Interpret Visualization
(a)
Interpret task/Form intention
Simulate Mental Visualization
Identify Relevant Data Dimensions
Conceive Construction Strategy
Execute Construction Operations
Interpret Visualization
(b)

Figure 22. Models of user interaction with Ploceus for visualization construction and analysis: Model (a) represents the hypothesized model and model (b) represents the revised version based on qualitative data.

Difficulties in stages of action. One participant (P3) appeared to have understood the features of Ploceus after the demo session and quickly constructed a visualization after we gave her the questions. Interestingly, she created a network connecting researchers with organizations and then did a projection on the organizations. When we asked her to describe what the visualization was showing, she realized that she was not creating a visualization that would answer the questions. She then created a different network in which researchers are connected to each other if they come from the same organizations, and again, this visualization was not able to answer the given questions. She finally realized what went wrong and commented that "I totally mis-interpreted the question." After careful consideration of the semantics of the questions, she successfully created the intended visualization. We interpret from this process that she had difficulty *interpreting what questions were being asked*.

Some users showed signs of a lack of ability to *mentally visualize the representation* that would answer a given question. One participant (P4) successfully created a one-mode network of collaborating organizations. To answer the question, "In which year(s) do we see the most cross-organization collaboration?," he added the Date column values as nodes to the visualization and connected the dates with the organizations. He tried hard to answer the question by examining the visualization, but did not have any viable lead.

In general, identifying the relevant data dimensions was not a major difficulty for most of the users. For example, to answer Q1 ("create a visualization showing the collaboration pattern between organizations on research grants"), all participants were aware that organizations and grants needed to be part of the visualization. Three participants (P4, P6, and P7), however, were uncertain if the name column in the Person table was relevant. All of them talked about their logic in the same way: organizations collaborated if researchers from these organizations had received grants together. In this sense, researchers were crucial components in the visualization. As a result, they tried to connect researchers with grants and to connect grants with organizations. This approach gave them no viable lead. After P7 finally constructed the correct visualization through multiple trials, he commented, "I didn't realize the system is so powerful that you can directly connect organizations with grants together!"

Even if users know clearly what variables are important, they may still experience potential difficulties to conceive the appropriate steps to construct the visualization. Previous studies have shown that visualization construction is a major hurdle for novice users.[58] In observing the participants, we got similar impressions. P4 wanted to create a one-mode network by projection, but he forgot to create the connections between the two classes of nodes first, and the projection could not be performed. In this case, he could not *conceive a proper sequence of the operations* to construct the visualization.

Most of the users did not have any difficulties in translating an intention of performing an operation to an actual action. That is, they were able to pinpoint the user interface component designed to support a specific operation. One participant (P4), however, could not figure out how to perform slicing 'n dicing, even though we demonstrated this functionality for him. He commented, "Although I've seen it, I just completely forgot about it." We acknowledge that the demonstration session was relatively short, and viewing a demonstration is very different from performing the same action oneself. However, the participant was confident that he would perform much better if he had experimented with Ploceus for a longer period of time.

All users were familiar with node-link diagrams and had no difficulty in reading the visualizations generated. The major difficulty surfaced when the size of the generated network became too large. The layout algorithms included in Ploceus could be improved. The force-directed layout, for example, was slow to stabilize for a network with more than 500 nodes, and it did not show clusters inherent in a network clearly. Disconnected subnetworks tended to be pushed to the boundaries of the visualization, making the graph less readable.

Where data do not fit model: trial-and-error strategy. While the preconceived action model anchored our interpretation of user difficulties in using Ploceus, upon analyzing user interaction history, we realized that the model might be an idealized sequence of visualization construction and exploration. In many cases, the participants' actions did not adhere to the presumed action sequence. The actual construction processes were quite ad hoc.

For example, while we did find evidence that could be interpreted as users mentally visualizing their desired representations (P7, for example, reported that he wanted to construct a visualization shown in the demo session, where a two-mode network was projected into a one-mode network), some participants did not mentally visualize the appropriate visualizations before constructing these visualizations. Instead, they talked about how they would "try it out" when they decided to perform an operation and commented they were not sure what the resulting visualizations would look like. This strategy could be due to unfamiliarity with Ploceus' working mechanism but could also be an effort of cognitive offloading. Rather than planning everything in the head, it was easier to put a thought into action. As P9 mentioned, "let me just try to see if this works, if not, I'll just start over."

As a consequence of adopting the trial-and-error strategy, the participants iterated between the stages in a nonlinear fashion, as shown in Figure 22(b). Participant P6, who failed to create appropriate visualizations within the given time frame, largely iterated in the loop of "interpret question/form intention → identify data dimensions ↔ execute construction operations ↔ interpret visualizations." A major cause of failure, as we interpreted, was that he was not able to mentally visualize an appropriate visualization and could not conceive a proper strategy to implement the visualization. As a result, he purely relied on randomly picking the operations.

Role of database knowledge

We speculated before the study that participants with background knowledge in database technologies, especially SQL queries, would understand the system better and perform the tasks with less difficulty. Qualitative evidence showed, however, this assumption was too simplistic.

Among the four participants who encountered no difficulty in completing the tasks, two had taken database classes, and the other two had no knowledge of SQL. Participant P8 was a software engineer and used SQL in his projects; yet, he spent a significant amount of time trying to understand the interface. P8 kept talking about operations he wanted to perform in terms of database concepts (e.g. "How do I do a GROUP BY?"). A major difficulty for him, as he described it, was to map the various SQL queries to the interactive operations supported in Ploceus. He even suggested that he would like to see a window showing the corresponding SQL queries whenever he interacted with the interface. It was also interesting to note that in answering Q3 ("In which year do we see the most collaboration between organizations"), although he observed that there were obviously more links in the "2003" subnetwork, he was not confident in giving a definite answer and commented that he would like to see a precise numeric value to confirm the answer. For a database expert like P8, whose thinking was deeply rooted in SQL queries, the learning curve might be higher than nonexpert users.

General impression and comments

All the participants liked Ploceus, especially the interface design. Although some participants considered constructing network visualizations nontrivial, they still agreed that the interface was consistent and the learning curve was not too high. The affordance of the network schema view was clear to all the participants, and all of them strongly favored this view.

The participants also identified features in Ploceus that they had difficulties in understanding and interpreting. More than one participant mentioned that the affordances of slicing 'n dicing shelves were not immediately clear to first-time users, but they acknowledged that after some interaction the design began to make sense for them. One participant also mentioned that the meaning of the numerical value following each slicing 'n dicing value was not clear. For example, when we slice 'n dice an organization collaboration network by year, a slice will be displayed as "2000 (282)," as shown in Figure 14, indicating the number of grants given in the year 2000.

One participant was so interested in Ploceus that he asked for some extra time after the given tasks to perform some open-ended exploration. One of the questions he wanted to know was who got the most money from NSF. He tried to create a network connecting researchers with amounts and then tried to order the amount nodes by value in the list view. While this is certainly one way to answer the question, a bar chart might be a more effective visual representation than a network visualization. In this regard, the user should have picked systems such as Tableau[22] instead of Ploceus. This observation is consistent with what Kobsa[59] has noted in his evaluation study of early InfoVis systems: users tend to use the default system or setting given to them, and it is difficult for them to initialize a change in the mindset to explore alternative representations or system settings.

Conclusion and future study

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

- Drawing from prior study, we present a conceptual framework specifying possible operations for constructing and transforming networks from multivariate tabular data. Most of these operations are meaningful to end users who are not necessarily database experts.
- 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 study on visual analytics.
- A qualitative evaluation that proposes a model of how users construct and explore network visualizations using Ploceus.

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 that explicitly illustrate the provenance of the data and integrating analytic techniques on data tables such as log-linear modeling on top of the network analysis techniques presented here. It also makes sense to provide a visual representation of users' interaction history. Such a construction history can be useful for nonexpert users to understand the consequences of their actions. As mentioned in section "Expressive power," 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. Since the user evaluation presented here involves controlled tasks for a specific dataset, we would also like to gain further insights on the system's ecological validity in longer-term case studies.

Funding

This study was 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. Chen PP. The entity-relationship model—toward a unified view of data. *ACM T Database Syst* 1976; 1(1): 9–36.
2. Elmasri R and Navathe S. *Fundamentals of database systems*. 6th ed. Boston, MA: Addison-Wesley, 2011.
3. Chen C, Yan X, Zhu F, et al. Graph OLAP: a multidimensional framework for graph data analysis. *Knowl Inf Syst* 2009; 21: 41–63.
4. Bagui S and Earp R. *Database design using entity-relationship diagrams*. 1st ed. Boston, MA: Auerbach Publications, 2003.
5. Amar RA and Stasko JT. Knowledge precepts for design and evaluation of information visualizations. *IEEE T Vis Comput Gr* 2005; 11(4): 432–442.
6. Lee B, Plaisant C, Parr CS, et al. Task taxonomy for graph visualization. In: *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, Venice, Italy, 23 May 2006, pp. 1–5. NY: ACM Press.
7. Shneiderman B and Aris A. Network visualization by semantic substrates. *IEEE T Vis Comput Gr* 2006; 12(5): 733–740.
8. Liu Z and Stasko J. Mental models, visual reasoning and interaction in information visualization: a top-down perspective. *IEEE T Vis Comput Gr* 2010; 16(6): 999–1008.
9. NetMiner—social network analysis software, http://www.netminer.com (2010).
10. Pajek—program for large network analysis, http://pajek.imfm.si/doku.php (2010).
11. UCINET, http://www.analytictech.com/ucinet/ (2010).
12. Adar E. GUESS: a language and interface for graph exploration. In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*, Quebec, Canada, 22-27 April 2006.
13. Hansen D, Shneiderman B and Smith MA. *Analyzing social media networks with NodeXL: insights from a connected world*. San Francisco, CA: Morgan Kaufmann, 2010.
14. Henry N, Fekete J and McGuffin MJ. NodeTrix: a hybrid visualization of social networks. *IEEE T Vis Comput Gr* 2007; 13(6): 1302–1309.
15. Perer A and Shneiderman B. Balancing systematic and flexible exploration of social networks. *IEEE T Vis Comput Gr* 2006; 12(5): 693–700.
16. Liu Z, Navathe S and Stasko J. Network-based visual analysis of tabular data. In: *IEEE conference on visual analytics science and technology '11*, Providence, RI, 23-28 October, 2011, pp. 41–50. Piscataway, NJ: IEEE Press.
17. Livny M, Ramakrishnan R, Beyer K, et al. DEVise: integrated querying and visual exploration of large datasets. *SIGMOD Rec* 1997; 26(2): 301–312.
18. Rao R and Card SK. 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*, Boston, Massachusetts, USA, 24-28 April 1994, pp. 318–322. Boston, MA: ACM.
19. Spenke M, Beilken C and Berlage T. FOCUS: the interactive table for product comparison and selection. In: *Proceedings of the 9th annual ACM symposium on user interface software and technology*, Seattle, WA, USA, 6-8 November 1996, pp. 41–50. New York, NY: ACM Press.
20. Spenke M and Beilken C. InfoZoom: analysing formula one racing results with an interactive data mining and visualisation tool. In: *Proceedings of 2nd international conference on data mining*, Cambridge University, Cambridge, UK, 5-7 July 2000, pp. 455–464. Southampton, UK: WIT Press.
21. Stolte C, Tang D and Hanrahan P. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE T Vis Comput Gr* 2002; 8(1): 52–65.
22. Tableau Software, http://www.tableausoftware.com/ (2010).
23. Stasko J, Görg C and Liu Z. Jigsaw: supporting investigative analysis through interactive visualization. *Inform Visual* 2008; 7(2): 118–132.

24. Heer J, Card SK and Landay JA. Prefuse: a toolkit for interactive information visualization. In: *Proceedings of the SIGCHI conference on human factors in computing systems*, Portland, Oregon, 2-7 Apr 2005, pp. 421–430. NY: ACM Press.
25. O'Madadhain J, Fisher D, White S, et al. *JUNG: Java universal network/graph framework*. Technical report UCI-ICS 03-17, 2003.
26. Gephi: an open source graph visualization and manipulation software, http://gephi.org (2010).
27. Auber D. Tulip: a huge graph visualization framework. In: Jünger M and Mutzel P (eds) *Graph drawing software (mathematics and visualization)*. Berlin: Springer-Verlag, 2003, pp. 105–123.
28. Freire M, Plaisant C, Shneiderman B, et al. ManyNets: an interface for multiple network analysis and visualization. In: *Proceedings of the 28th international conference on human factors in computing systems*, Atlanta, GA, 10-15 April 2010, pp. 213–222. NY: ACM Press.
29. Wattenberg M. Visual exploration of multivariate graphs. In: *Proceedings of the SIGCHI conference on human factors in computing systems*, Montreal, Quebec, 22-27 April 2006, pp. 811–819. New York: ACM.
30. Guéhis S, Rigaux P and Waller E. Data-driven publication of relational databases. In: *10th international database engineering and applications symposium*, Delhi, India, 11-14 December 2006, pp. 267–272. Piscataway, NJ: IEEE Press.
31. Bohannon P, Korth HF and Narayan PPS. The table and the tree: on-line access to relational data through virtual XML documents. In: *Proceedings of WebDB*, Santa Barbara, California, USA, 24-25 May 2001, pp. 55–60. NY: ACM.
32. Wilkinson L. *The grammar of graphics*. NY: Springer-Verlag, 2005.
33. Weaver C. Multidimensional data dissection using attribute relationship graphs. In: *IEEE symposium on visual analytics science and technology (VAST)*, Salt Lake City, Utah, 24-29 October 2010, pp. 75–82. Piscataway, NJ: IEEE Press.
34. TouchGraph Navigator: graph visualization and social network analysis software, http://touchgraph.com/navigator (2011).
35. Centrifuge systems, http://centrifugesystems.com/ (2011).
36. Heer J and Perer A. Orion: a system for modeling, transformation and visualization of multidimensional heterogeneous networks. In: *IEEE conference on visual analytics science and technology '11*, Providence, RI, 23-28 October, 2011. Piscataway, NJ: IEEE Press.
37. Weaver C. Cross-filtered views for multidimensional visual analysis. *IEEE T Vis Comput Gr* 2010; 16(2): 192–204.
38. North C, Conklin N, Indukuri K, et al. Visualization schemas and a web-based architecture for custom multiple-view visualization of multiple-table databases. *Inform Visual* 2002; 1(3–4): 211–228.
39. Dijkstra EW. A note on two problems in connexion with graphs. *Numer Math* 1959; 1: 269–271.
40. Latapy M, Magnien C and Vecchio ND. Basic notions for the analysis of large two-mode networks. *Soc Networks* 2008; 30(1): 31–48.
41. Cypher A and Halbert DC. *Watch what I do: programming by demonstration*. Cambridge, MA: The MIT Press, 1993.
42. Fruchterman TM and Reingold EM. Graph drawing by force-directed placement. *Software Pract Exper* 1991; 21(11): 1129–1164.
43. Kamada T and Kawai S. An algorithm for drawing general undirected graphs. *Inform Process Lett* 1989; 31(1): 7–15.
44. Meyer B. Self-organizing graphs: a neural network perspective of graph layout. In: Sue Whitesides (ed.) *Proceedings of the 6th International Symposium on Graph Drawing (GD '98)*, Montreal, Canada, 13-15 August 1998, pp. 246-262. London, UK: Springer-Verlag.
45. Cleveland WS and McGill R. Graphical perception and graphical methods for analyzing scientific data. *Science* 1985; 229(4716): 828–833.
46. Mackinlay J. Automating the design of graphical presentations of relational information. *ACM T Graphic* 1986; 5(2): 110–141.
47. Mackinlay JD, Hanrahan P and Stolte C. Show me: automatic presentation for visual analysis. *IEEE T Vis Comput Gr* 2007; 13(6): 1137–1144.
48. Ware C. *Visual thinking for design*. Burlington, MA: Morgan Kaufmann, 2008.
49. Ware C. *Information visualization: perception for design*. San Francisco, CA: Morgan Kaufmann, 2004.
50. Smith MA, Shneiderman B, Milic-Frayling N, et al. Analyzing (social media) networks with NodeXL. In: *Proceedings of the fourth international conference on communities and technologies*, University Park, PA, USA, 25-27 June 2009, pp. 255–264. University Park, PA: ACM.
51. GraphML file format, http://graphml.graphdrawing.org/ (2010).
52. Codd EF. A relational model of data for large shared data banks. *Commun ACM* 1970; 13(6): 377–387.
53. NetBeans rich-client platform development, http://netbeans.org/features/platform/ (2011).
54. H2 database engine, http://www.h2database.com/html/main.html (2011).
55. Meijer E and Bierman G. A co-relational model of data for large shared data banks. *Queue* 2011; 9(3): 30–48.
56. Van Ham F and Perer A. "Search, Show Context, Expand on Demand": supporting large graph exploration with degree-of-interest. *IEEE T Vis Comput Gr* 2009; 15(6): 953–960.
57. Norman DA. *The design of everyday things*. NY: Basic Books, 2002.
58. Grammel L, Tory M and Storey M. How information visualization novices construct visualizations. *IEEE T Vis Comput Gr* 2010; 16(6): 943–952.
59. Kobsa A. An empirical comparison of three commercial information visualization systems. In: *IEEE symposium on information visualization*, San Diego, California, 22-23 October 2001, pp. 123–130. Piscataway, NJ: IEEE Press.