Visual Analysis of Higher-Order Conjunctive Relationships in Multidimensional Data Using a Hypergraph Query System

Rachel Shadoan and Chris Weaver, *Member, IEEE Computer Society*

Abstract—Visual exploration and analysis of multidimensional data becomes increasingly difficult with increasing dimensionality. We want to understand the relationships between dimensions of data, but lack flexible techniques for exploration beyond low-order relationships. Current visual techniques for multidimensional data analysis focus on binary conjunctive relationships between dimensions. Recent techniques, such as cross-filtering on an attribute relationship graph, facilitate the exploration of some higher-order conjunctive relationships, but require a great deal of care and precision to do so effectively. This paper provides a detailed analysis of the expressive power of existing visual querying systems and describes a more flexible approach in which users can explore n-ary conjunctive inter- and intra- dimensional relationships by interactively constructing queries as visual hypergraphs. In a hypergraph query, nodes represent subsets of values and hyperedges represent conjunctive relationships. Analysts can dynamically build and modify the query using sequences of simple interactions. The hypergraph serves not only as a query specification, but also as a compact visual representation of the interactive state. Using examples from several domains, focusing on the digital humanities, we describe the design considerations for developing the querying system and incorporating it into visual analysis tools. We analyze query expressiveness with regard to the kinds of questions it can and cannot pose, and describe how it simultaneously expands the expressiveness of and is complemented by cross-filtering.

Index Terms—Graph search, graph query language, multidimensional data, attribute relationship graphs, multivariate data analysis, higher-order conjunctive queries, visual query language, digital humanities

1 Introduction

Relationships between attributes in multidimensional data are of central interest in data analysis, but identifying and characterizing these relationships becomes increasingly onerous with increasing dimensionality. Flexible representations are needed to facilitate access and exploration of patterns in co-occurrences of attribute values across dimensions. Many techniques in multidimensional data analysis can express questions only about low-order relationships between dimensions of data, typically two or three at a time. More recent techniques provide the capability to express questions about larger numbers of dimensions; however, the process is often tedious and difficult to conceptualize. For instance, cross-filtering [28] can express questions involving more than four dimensions of data, but the process is tedious and difficult to conceptualize.

What is needed is a flexible system to construct higher-order queries that makes the questions being asked explicit. Graphs are a natural choice to represent the relationships in data, as they conform well to the associative models we use to organize information. There has been much interest in graph representations in recent years; in addition to homogeneous graphs representing things like social networks, applications and techniques using heterogeneous graph representations have begun to garner attention for their ability to show the relationships between different classes of data items. Indeed, there are numerous graph querying languages in the literature that provide the ability to construct higher-order conjunctive queries about the relationships between data items having certain attributes. While these languages are well-suited to information retrieval, they are of limited utility for multidimensional data analysis because they do not facilitate the direct exploration of relationships between attributes.

Attribute relationship graphs [29] do facilitate the direct exploration of relationships between attributes. Cross-filtering on an attribute relationship graph allows exploration of many-to-many binary conjunctive relationships between dimensions of data, but does not accurately visually reflect ternary and higher-order conjunctive relationships. We would like to develop a tool that maintains the expressiveness of cross-filtering and leverages the power of the attribute relationship graph, but that also allows for queries to be developed more flexibly and easily.

We describe a flexible interactive query language for explicitly constructing queries involving n-ary conjunctive inter- and intra-dimensional relationships, applicable to categorical and categorically-reducible dimensions in multidimensional data sets. It contributes:

- a detailed analysis of the kinds of relationships expressible using cross-filtering and cross-filtering on an attribute relationship graph;
- a modular design for creating interactive visual graph querying systems;
- a hypergraph querying language that can express n-ary conjunctive inter- and intra-dimensional many-to-many relationships;
- an implementation of a hypergraph querying system;
- two exemplar visual tools for exploring and analyzing complex multidimensional data sets using a hypergraph querying system.

We begin by describing an example in which the hypergraph query technique is used for visual exploration of relationships in the Electronic Enlightenment database. After discussing the multidimensional analysis techniques that this technique builds on, we describe the hypergraph query technique and the considerations that went into its design, as well as presenting an implementation of the technique. Using additional examples, we demonstrate the expressiveness of the technique. We close with directions for future work.

2 Example: Candid

The Electronic Enlightenment [24] is a wide-ranging online collection of correspondence from the early modern period. The EE data set includes over 58,000 letters sent during the Enlightenment, in addition to information about the more than 7000 letter senders and recipients. The database contains a variety of details about the letters, including the primary language the letter is written in; the letter's source and

- *Rachel Shadoan is with Akashic Labs LLC. E-mail: rachel.shadoan@gmail.com.*
- *Chris Weaver is with the School of Computer Science and the Center for Spatial Analysis at University of Oklahoma. E-mail: weaver@cs.ou.edu.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

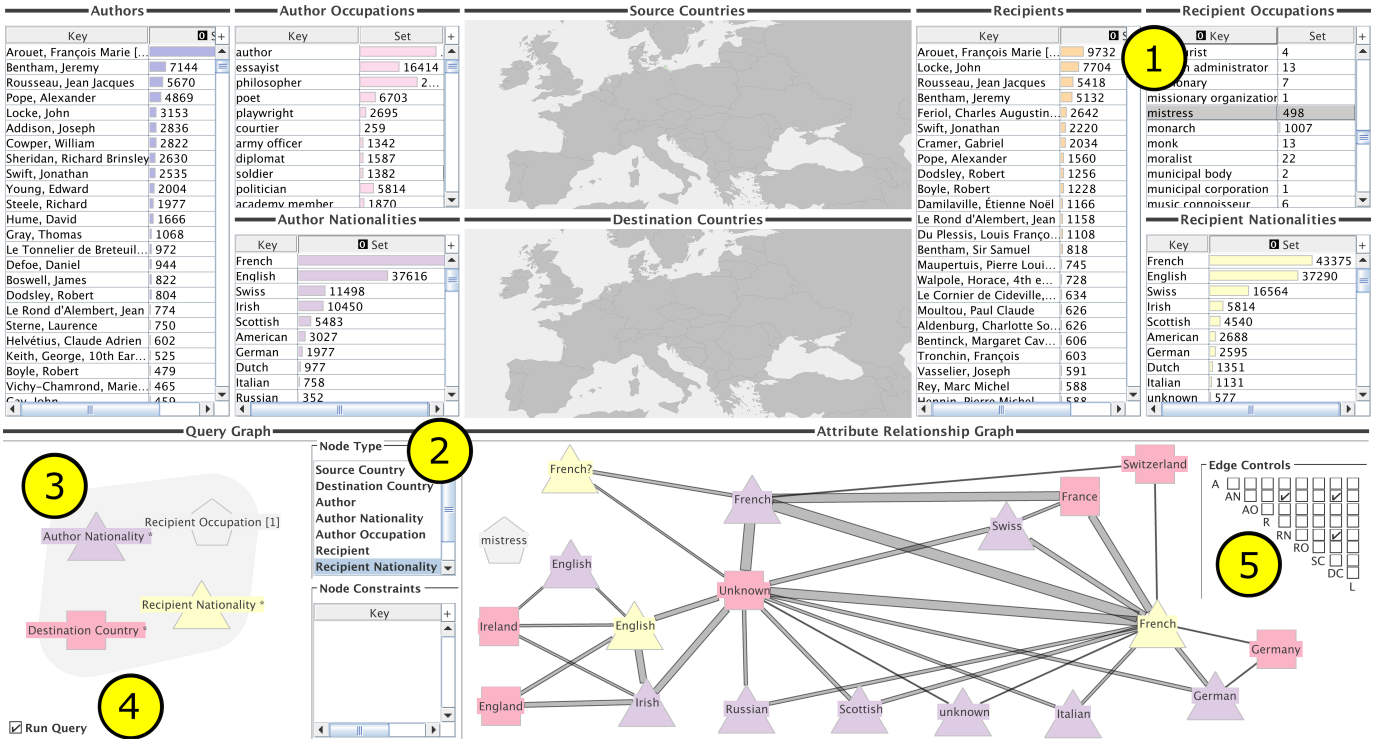


Fig. 1. Candid, displaying the results of a query regarding the nationality and location of mistresses in the Electronic Enlightenment data set.

destination city, state, and country; the date the letter was sent; the nationality, birth place and date, death place and date, and occupations of the authors and recipients; and the age of the author and recipient at the time a letter was sent, to name most. This large and complex data set contains a multitude of many-to-many relationships, which are difficult to explore with traditional search functions.

We used Improvise [26] to develop Candid, an interactive visual tool for digital humanities scholars to explore and analyze the relationships between attributes in the Electronic Enlightenment data set. The tool, shown in Figures 1–3, maps seven attributes of the letters database—letter author, letter recipient, primary language of letter, author/recipient nationality, author/recipient occupation, letter source country, and letter destination country—into a collection of multiple coordinated views laid out in a single-window user interface. The attribute values populate table views, from which the user can select a desired subset of values to add to nodes in sculpting queries, which are built in the small graph view. Each node in the query graph matches a subset of nodes in the attribute relationship graph. Binding the nodes in the query graph together in a hyperedge further restricts the values that appear in the attribute relationship graph [29]. The attribute relationship graph, filtered on the results of the query, shows the relationships between letter attributes; if an edge exists between two nodes in the attribute relationship graph, there is at least one letter having the attributes represented by the nodes connected by the edge. Thicker edges represent more letters.

Figure 1 shows an example of one point in an exploration of correspondence, investigating the locations and nationalities of individuals who have the occupation "mistress", as well as the nationality of those who wrote to the mistresses. Phrased another way, this example asks, *"What nationality were the mistresses in this data? Where were these mistresses receiving letters, and from what nationality of author?"* This query is constructed as follows. After selecting the recipient occupation "mistress" (1), and adding a node constrained to that value to the query graph, the user adds nodes representing any recipient nationality, author nationality, or destination country to the query graph (2). Once all of those nodes have been added to the query graph, the user adds each of the nodes to a hyperedge (3) before running the query (4). When the query has returned, the user adjusts the edges (5), revealing that French mistresses were perhaps the most internationally popular in this data set, receiving letters from writers of seven different nationalities, though there are more letters between French authors and French mistresses than any of the other nationalities. English mistresses, by comparison, received letters from writers of only two nationalities: English and Irish. Additionally, we can see that while the destination of a significant number of the letters were received by mistresses are unknown, a large number of the letters received by French mistresses were received in France. A smaller number of letters received by French mistresses were received in Germany and Switzerland. English mistresses, by comparison, received letters only in England and Ireland. As one might expect, given the somewhat tense relationship between France and England during the Enlightenment, no letters in the data set went from English authors to French mistresses (though perhaps, given the apparent popularity of the French mistresses, the English were missing out!)

If a user wanted to dig into this topic further, perhaps uncovering more about these internationally popular French mistresses, she could do so by constraining the recipient nationality node in the query graph to French, and adding unconstrained nodes for Author and Recipient to the current hyperedge. The results of that query would reveal the identities of the French mistresses and their correspondents.

3 Related Work

Commonly, node-link (network) diagrams are used to show the relationships between dimensions in nominal data. General solutions for visualizing networks include toolkits like prefuse [10] and JUNG [15], which provide network visualization capabilities that can be incorporated into analysis tools, and applications like Pajek [1] and Visone [4], which provide analysis capabilities for a given network.

There are a variety of paradigms for leveraging graphs for multidimensional data analysis. PivotGraph [25], semantic substrates [19], GraphDice [2], and MatrixExplorer [11] use location to distinguish between attributes. Semantic abstraction techniques, featured in OntoVis [16] and Ploceus [13], use the underlying semantics of the graph to abstract data for easier analysis. Jigsaw [20], cross-filtering [29],

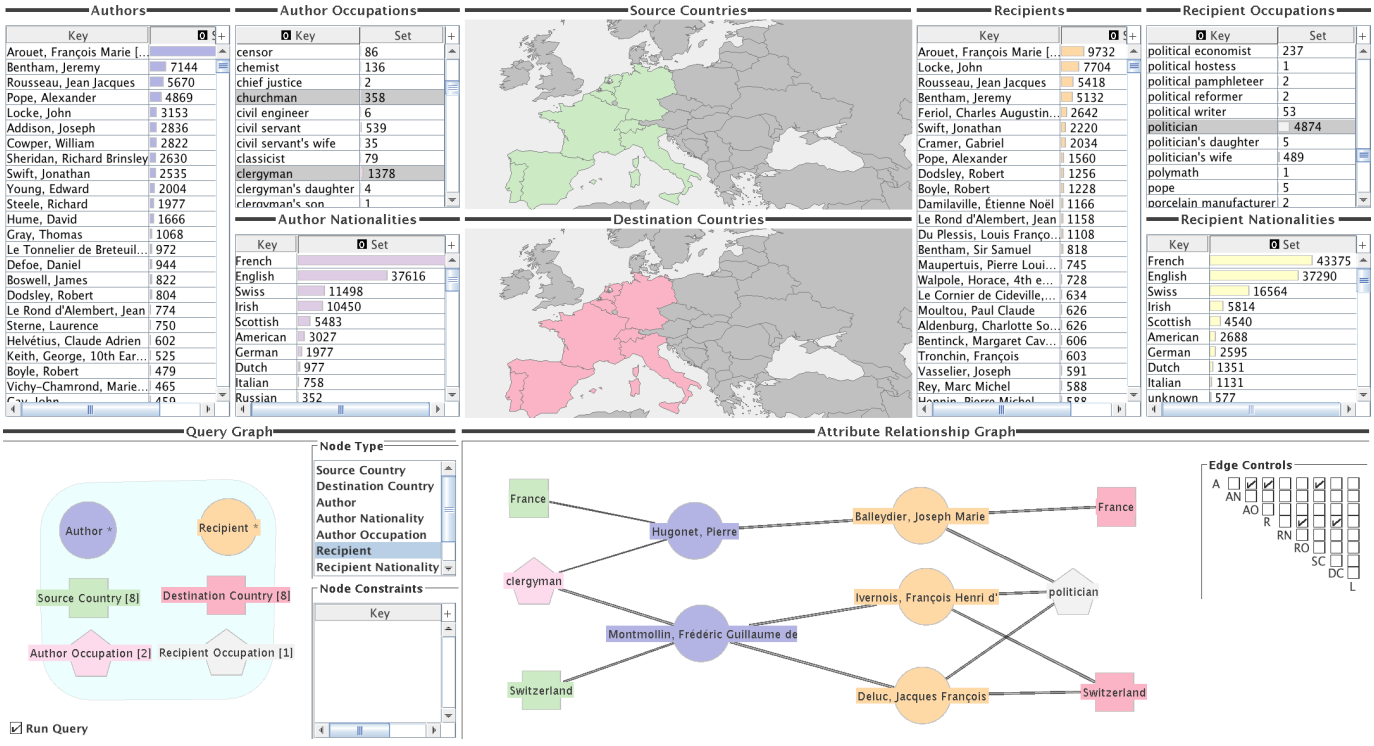


Fig. 2. Candid, displaying the results of a query about the political activity of clergy and churchmen in Europe. A higher-order inter-dimensional conjunctive query expresses the question *"Which clergy or churchmen in continental Europe wrote to which politicians?"*. *Author Occupation* is constrained to clergy or churchman; *Recipient Occupation* is constrained to politician; *Source Country* and *Destination Country* are both constrained to France, Germany, Belgium, Italy, Switzerland, Portugal, Spain, and the Netherlands. Two members of the clergy, Pierre Hugonet and Frederic Guillaume de Montmollin of France and Switzerland, respectively, were writing to three politicians: Joseph Marie Balleydier in France and Francois Henri d' Ivernois and Jacques Francois Deluc of Switzerland. Interestingly, the clergyman in France was only writing to the politician in France, and the clergyman in Switzerland was only writing to the politicians in Switzerland.

and Ploceus use select-and-filter techniques.

Jigsaw [20] is an application designed for intelligence analysis that combines a variety of simple views, including tables and a node-link (graph) diagram, for exploration of relationships between entities in documents collected for analysis. Users may add lists of desired entities and explore the relationships between them by selecting subsets, which highlights the related items in other lists of entities. However, the graph view does not support direct exploration of attribute relationships; that exploration must be done by selecting entities or documents of interest in the graph and investigating them in other views.

Cross-filtering [28] is a pattern for composing multiple-coordinated views to facilitate the dissection of interdimensional relationships. In a cross-filtering visualization, views are connected by a framework of set queries which allow the user to filter one view based on the values selected in another view. By chaining together selecting and filtering actions, users can explore relationships between subsets of attribute values across multiple dimensions of data. In combination with an attribute relationship graph, cross-filtering can be used to explore many-to-many binary conjunctive relationships. Attribute relationships graphs [29], unlike attributed graphs, allow for the direct exploration of relationships between attributes.

Ploceus [13] allows users to dynamically construct networks from tabular data. It is similar to cross-filtering on an attribute relationship graph in form and in the kinds of questions it can answer, but it provides additional flexibility by allowing users to define the semantics by which the nodes in the graph are linked. Additionally, it provides semantic abstraction functionality for creating higher-order graphs.

While similar to cross-filtering on an attribute relationship graph and Ploceus, the hypergraph query technique described in this paper differs in explicitness and expressiveness. Queries on the data are visualized *explicitly*. In the techniques discussed above, the query being expressed exists implicitly in the interactive state of the visualization. Different parts of the query may be spread out across many different views and other control surfaces. Explicitly constructing and visualizing the query frees the user from having to remember it or interpret it from the entire interactive state. The hypergraph query language is also able to express higher-order conjunctive queries that are difficult or impossible to express using previous techniques.

4 Limitations of Existing Approaches

The purpose of the hypergraph query technique described in this paper is to provide analytic utility not provided by prior visual querying systems. Before describing the technique, however, it is necessary to discuss both the way we classify relationships in data and the systems that informed its design. In particular, we will look at cross-filtering [28], both alone and on an attribute relationship graph [29], focusing on the kinds of relationships that technique can explore. Once the space of questions expressible by that technique is established, we describe the space of questions we would like to be able to explore.

4.1 Describing Relationships in Data

Relationships between data attributes can be described in terms of their *logical basis*, their *degree*, and their *connectivity* [21].

The *logical basis* of the relationship refers to the logical association between the attributes in the relationship. Relationships can be conjunctive, meaning that the attributes in that relationship are associated by a logical *AND*. Conjunctive relationships are "both" or "all" relationships. Relationships can also be disjunctive, meaning that the attributes in that relationship are associated by a logical *OR*. Disjunctive relationships are "either" or "any" relationships.

The *degree* of a relationship is the number of attributes associated with that relationship. If there is only one attribute associated with a relationship, we refer to that relationship as *unary*. Relationships with two attributes are referred to as *binary* relationships, with three

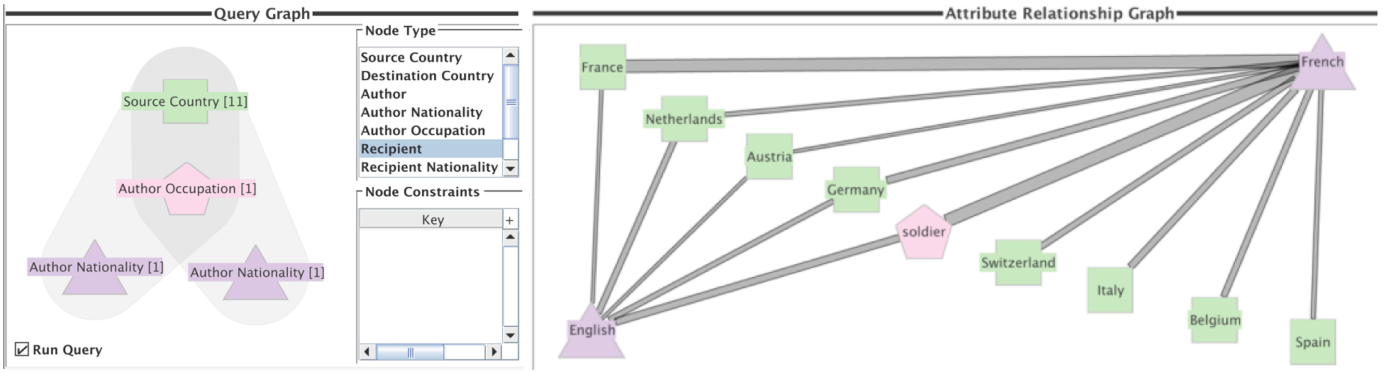


Fig. 3. Movement of French and English soldiers in continental Europe. A disjunctive query between two ternary inter-dimensional conjunctive sub-queries expresses the question: *"From which countries in continental Europe did French and English soldiers send letters?" Author Nationality* on the left is constrained to French; *Author Nationality* on the right is constrained to English; *Author Occupation* is constrained to soldier; *Source Country* is constrained to French, Germany, Belgium, Switzerland, Portugal, Spain, the Netherlands, Austria, Poland, Italy, and the Czech Republic. *Author* and *Recipient* are unconstrained and can match any value. French soldiers write more letters in general than English soldiers. They also write letters from more countries than English soldiers. Interestingly, while French soldiers write letters from France, there are no letters from English soldiers in England in the data set.

attributes as *ternary* relationships, and so on. In this paper we refer to anything involving more than two attributes as *higher-order* relationships. Table 1 shows examples of relationships of various degrees.

The *connectivity* of a relationship specifies the mapping between attributes in the relationships. Connectivity is described as "one" or "many", though the actual number describing the "many" is called the *cardinality* of the connectivity. Examples of one-to-one, one-to-many, and many-to-many relationships are shown in Table 2.

Multidimensional data analysis is a process of looking at relationships between subsets of data dimensions in order to understand that data as a whole. Thus, for the purpose of multidimensional data analysis, we are interested in the kinds of relationships that relate dimensions in meaningful ways. Unary relationships, for instance, hold interest in as much as they reveal patterns of distribution of data item counts within a dimension, but they do not provide information about relationships between dimensions. Disjunctive relationships are similarly of typically low interest; a disjunctive relationship between attributes corresponds to the join of those attributes. The existence of attribute values in a disjunctive relationship does not imply anything about the relationship between those two attributes beyond that both attribute values exist in the data set. It is only when combined with conjunctive relationships that disjunctive relationships become useful, as they can be used to compare conjunctive relationships.

Conjunctive binary relationships, as the lowest-order relationship to provide significant utility for analysis of connection, have been the focus of much work in visual analytics. Ternary and higher-order conjunctive relationships are often encoded in visualization techniques that allow for direct comparison between pairs of attributes, but those relationships are still explored almost entirely implicitly, through visual representations, rather than explicitly, through interactive querying. Recent interaction techniques, such as in Jigsaw [20] and cross-filtering [28], allow explicit exploration of binary conjunctive relationships, but are limited in terms of the connectivity of higher-order conjunctive relationships that can be expressed and thus explored.

Table 1. Relationship Degree

Degree	Example	Value	Description
Unary (1)	Author Name	Voltaire	Voltaire wrote letters
Binary (2)	Author Name, Language	Voltaire English	Voltaire wrote in English
Higher-order (3+)	Author Name, Language, Source Country, Destination	Voltaire English France England	Voltaire wrote in English from France to England

Table 2. Relationship Connectivity

Connectivity	Example	Description
One-to-one	Person to Birth Date	A person is born on one date
One-to-many	Person to Language	A person may speak many languages
Many-to-many	Language to Country	Many languages may be spoken in a given country, and many countries speak the same language.

4.2 Cross-Filtering

Cross-filtering is a design pattern for composing multiple coordinated views to facilitate multidimensional data analysis. To form complex queries, users perform a sequence of filtering and selection actions on views populated with various dimensions of the data.

A cross-filtering visualization is built on a framework of queries. First, the data records being visualized are partitioned by the unique attribute values for each of the dimensions being included in the cross filtering. A dimension with five unique attribute values produces five groups of data records, indexed by the attribute values, each containing the data items that have that attribute value. These *groups* are used to populate views for each of the dimensions. Views allow selection of subsets of values as part of the querying process. When an attribute value from one dimension is selected, and a second view is filtered on that selection, the items shown in the second view are the attribute values present in the groups of data records indexed by the attribute values selected in the first view.

Cross-filtering is a powerful technique for multidimensional data analysis, but it is primarily useful to explore one-to-one or one-to-many binary conjunctive relationships. It is possible to express some higher-order conjunctive relationships, but the process is interactively convoluted. For an n-ary conjunctive query, each dimension must be filtered on the selections in all of the previous dimensions.

Using cross-filtering to analyze many-to-many relationships is similarly difficult. Selecting multiple items in a dimension that another view is filtered on will show attribute values corresponding to *any* of the items selected in the first view. Thus, in order to analyze many-to-many relationships using cross-filtering, it is necessary to select one item, see the corresponding attribute values in the filtered view, select

a second item, and then mentally compare the attribute values corresponding to the second item selected with those that appeared for the first item. This connectivity restriction applies to higher-order relationships, as well. For an n-ary relationship, where the nth dimension is being filtered on $n-1$ dimensions, only one value may be usefully selected in each of the $n-1$ dimensions.

While it is possible (but difficult) to use cross-filtering to explore many-to-many relationships and some higher-order conjunctive relationships, there are higher-order conjunctive relationships it is unable to express. It is limited to queries in which the relationships are *between* dimensions. It is not possible to use cross-filtering to build a query in which two attributes in the relationship come from the same dimension. We may be able to construct a query that will coincidentally provide us with the answer, but it will also provide a lot of extra information that must be sifted through in order to locate the desired results. Not only is this inefficient, it can be visually overwhelming.

The difficulty in exploring many-to-many relationships is alleviated by using cross-filtering in combination with an attribute relationship graph, discussed in the following section. The inability to express conjunctive relationships within dimensions is addressed by the hypergraph querying technique described in this paper.

4.3 Cross-Filtering on an Attribute Relationship Graph

Attribute relationship graphs are an undirected graph representation in which the nodes represent the unique attribute values in the data set. If two attribute values co-occur in a data record, they are connected by an edge in the attribute relationship graph. It is distinct from attributed relational graphs [8], in which data items are connected to nodes by directed edges and attribute nodes are not directly connected to each other.

Attribute relationship graphs allow for direct exploration of sets of binary inter-attribute relationships. While homogeneous and heterogeneous attributed graphs facilitate the direct exploration of relationships between objects, but allow only implicit exploration of the relationships between the objects' attributes. In an attribute relationship graph, the values of each dimension are connected to the values in every other dimension with which they co-occur, so it is straightforward to compare any pair of dimensions.

Alone, however, an attribute relationship graph is visually overwhelming. The sheer number of nodes (one for each unique attribute value) and edges (one for each unique pair of attribute values) makes it difficult to navigate and comprehend. Cross-filtering provides a technique to query the attribute relationship graph; only the attribute values selected as part of a cross-filtering query appear as nodes, and only the edges that connect two of the nodes present in the graph are visible.

Visualizing the results of a cross-filtering interaction as an attribute relationship graph allows an analyst to see the many-to-many relationships between selected attribute values, expanding the set of questions that cross-filtering can ask. With an attribute relationship graph, cross-filtering can represent queries about many-to-many binary conjunctive relationships. However, the attribute-relationship graph creates some visual difficulty with the higher-order conjunctive relationships that cross-filtering can represent when used alone. The edges in an attribute relationship graph come from *cliques*. Cliquing, which is a grouping equijoin, groups the data into sets indexed by unique pairs of attribute values. This relationship is a *co-occurrence* relationship; the set of data items indexed by two attribute values contains all data records where those two attribute values both appear. Every unique pair of attribute values that occurs in the data set is connected by an edge. Those edges are not filtered based on higher-order relationships, even if the nodes in the graph are. The only guarantee about an edge between two nodes is that it represents the set of data items in which those attribute values co-occur. Consequently, even if the query involves a higher-order conjunctive relationship, the edges in the attribute relationship graph do not. This shortcoming is addressed by the technique described in this paper.

4.4 Desirable Relationships to Express

Cross-filtering, and cross-filtering on an attribute relationship graph, are powerful techniques for exploring relationships between attributes in multidimensional data. Cross-filtering provides a means to explore one-to-many binary conjunctive relationships and one-to-many higher-order inter-dimensional conjunctive relationships. With an attribute relationship graph, cross-filtering facilitates exploration of many-to-many binary conjunctive relationships. However, the techniques cannot express queries involving intra-dimensional conjunctive relationships, and do not accurately visually represent higher-order conjunctive queries. Hypergraph queries address these shortcomings.

5 System Goals

Query Expressiveness The primary goal of the technique described in this paper is to enable users to express queries that are not expressible using existing techniques, without losing the expressiveness provided by those existing techniques. Cross-filtering on an attribute relationship graph can express unary, binary, and higher-order one-to-one, one-to-many, and many-to-many conjunctive *inter-dimensional* relationships. Our technique makes it possible to also express queries about unary, binary, and higher-order one-to-one, one-to-many, and many-to-many conjunctive *intra-dimensional* relationships. In some cases, it is possible to formulate a cross-filtering query that will coincidentally provide answers to a question involving conjunctive intra-dimensional relationships. However, such queries will result in additional information that does not conform to the conjunctive intra-dimensional relationship. The results that do not conform to the conjunctive intra-dimensional relationship must be sifted through, often by way of tedious interaction, to isolate the results that conform to the desired relationship. The ability to express conjunctive intra-dimensional relationships allows the user to specify queries that will return only the intended results.

Query Explicitness Explicitly visualizing a complete query in a single location in the visualization frees the user from having to conceptualize and recall the entire interactive state of the visualization. It reduces the likelihood that the user will unintentionally alter the query, and thus increases the likelihood that the query will provide the results that the user expects.

Dynamic Query Construction Dynamic query construction, in which results appear as the user constructs a query, allows the user to make alterations in response to the results of a partially formed query. This provides a way for users to tailor queries to more closely match their intentions and thus provide results that match those expectations.

Exact Query Matching Providing only exact matches to queries maintains the link between the user's intentions and the desired outcome. A query reflects a question that the user is trying to find the "answers" to; if they ask a specific question by way of a query, the results should "answer" only that question so that the results are meaningful to the user.

Exhaustive Query Matching The mechanism that generates the results of queries provides every matching result in the data set. This allows users to see individual results in context of all of the results, facilitating the recognition of patterns.

Attribute Relationship Graph Compatibility Displaying query results on an attribute relationship graph facilitates the direct exploration of many-to-many relationships between attributes. However, it does not provide accurate visualization of higher-order conjunctive relationships. Filtering edges to accurately reflect higher-order conjunctive relationships allows us to substantially improve an already useful tool while more closely matching query results to a user's intention.

Acceptable Speed for Interaction The extended analytic utility of the querying system will be reduced if it is so slow that analysts will not use it. Ideally, query results would appear within 100 milliseconds of a user's interaction with the query, as that speed facilitates dynamic query adjustment [18]. If that speed is not attainable, queries running in a second or less would also be within typically human limits of

cognitive attention; querying times of a few seconds to a few minutes are not ideal for maintaining attention, and anything longer than a few minutes is by definition unacceptable in an interactive context.

6 System Design

This section describes a system that provides an explicit querying graphical user interface to flexibly express higher-order intradimensional queries. It then describes how the technique duplicates and extends the query expressiveness of cross-filtering on an attribute relationship graph. We then outline system design considerations.

The system functions as follows. A query is explicitly and interactively constructed in the visualization, an algorithm locates data items that match the query, and the results are displayed on an attribute relationship graph. This section details design considerations for the interactive visual query language, the data representation, and the query matching algorithm.

6.1 Visually Representing Relationships

Techniques like cross-filtering are visual query languages, in that they encode the query within the entire interactive state of the visualization. However, as the interactive state of a multiple-view visualization can be large and complex, it is difficult to visualize and recall a query in that form. Visual graph query languages explicitly visualize a query in a way that is easier for novice users to manipulate, interpret, and recall. The numerous examples of graph query languages in the literature (e.g., [6, 9, 17, 3, 5]) are informed by the structure of the graphs designed to be queried.

Existing graph query languages like QGraph [3] and GraphLog [6] use heterogeneous attributed graphs consisting of nodes and directed edges. Nodes are typed, and assigned attributes to restrict the data items that will match that node. If a path exists from one node to another in the graph, a conjunctive relationship exists between those nodes and all of the nodes along the path. Those languages do not provide a way to express disjunctive relationships between conjunctive queries, as one can in cross-filtering. However, one could imagine expressing this as a partitioned graph.

There are several problems with using a directed graph language for this application, however. The relationships underlying edges in an attributed relationship graph are undirected. This is the easiest of the problems to remedy; the graph language could simply use undirected edges. However, that creates an inconsistency between the meaning of connected subgraphs in the attribute relationship graph and the query graphs. Three nodes connected in the attribute relationship graph do not have a conjunctive relationship; conjunctive relationships in an attribute relationship graph are restricted to two nodes. Furthermore, visually, nodes directly connected by edges seem more connected than nodes that are in the same partition but not directly connected by an edge. Thus, the conjunctive relationship between the nodes in a query graph is visually ambiguous unless it is fully connected by edges.

A different solution is to use a hyperedge, drawn as a blob or shell around nodes, to represent conjunctive relationships. Disjunctive relationships would exist between any subgraphs not included in the same hyperedge. This makes the conjunctive relationship between nodes more apparent than it would be in partially connected graphs used in languages like QGraph, as in a hyperedge the relationship between all of the nodes is symmetrical.

6.2 Building Queries

Several interactive functions are required to construct a query, regardless of which query representation is used. Required functions are:

- Create a node. Node creation has two cases, *unconstrained* and *constrained*. In the unconstrained case, we only want to specify the dimension of the node, but not the specific attribute values that it is allowed to match. In the constrained case, we want to specify both the dimension of the node *and* the attribute values that it is allowed to match. Mechanisms must be included to specify node type and to indicate the attribute values that a node can match.
- Browse a node's attribute values. Displays the attribute values that a node is allowed to match.
- Delete a node. Removes a node from the query.
- Create a conjunctive relationship between selected nodes. Creates a conjunctive relationship between selected nodes.
- Delete a conjunctive relationship. Deleting a conjunctive relationship should not delete all nodes that the relationship contains.

There are other functions that, while not strictly necessary, can be provided for sake of convenience. These nice-to-have functions are:

- Create a node in an existing conjunctive relationship.
- Update an existing node's attribute values. This function allows a user to tweak the constraints on a node in the query without having to delete and re-add it.

6.3 Finding Query Results

Representing the query as a graph seems to lend itself to finding query results using a subgraph matching algorithm. However, there are a few reasons why, in practice, that may be less desirable.

First, subgraphs in an attribute relationship graph do not represent conjunctive relationships. In an attribute relationship graph, the only conjunctive relationships specified by the structure are pairwise, between the two nodes connected by an edge. Directly matching a query graph on an attribute relationship graph will not provide results that have the conjunctive relationships specified by the query. If subgraph matching is the desired query-finding mechanism, additional transformation of the attribute relationship graph is required in order to provide the conjunctive relationships specified by queries.

Second, exact, exhaustive subgraph matching algorithms are not fast for large graphs. GRay [22], the algorithm used by the Graphite tool [5], is a fast algorithm for pattern matching in large graphs, but it does not guarantee that the results returned will exactly match the query, nor that all matching subgraphs will be returned. Ullman [23], Nauty [14], and VF2 [7] provide exact subgraph matching, but both Ullman and Nauty place restrictions on the kinds of graphs they can match. Even with these algorithms, which are the state of the art, the time necessary to find matching subgraphs in large graphs may prove interactively prohibitive.

Fortunately, subgraph isomorphism algorithms are not our only options for finding query results. Most visual graph query languages form a skin over a simple query language like SQL. Transforming query graphs into standard database or table queries is a viable approach. Linear table scans, in which the query is evaluated against each record, are another option. Clever indices into tables, such as indices mapping each unique attribute value to the set of records that contain that attribute value, can improve the time complexity of table-scan based query evaluation.

7 Implementation

In this section, we present an implementation of the querying system discussed in Section 6. First, we consider the components of a query graph and the interactive scaffolding necessary to support user construction of query graphs. Then, we discuss the algorithm used to match the queries. Finally, we discuss the visual tools we implemented that incorporate this querying scheme.

7.1 Hypergraph Querying Language

A query graph is an attributed hypergraph consisting of attributed nodes and hyperedges. Nodes have an attribute type, which corresponds to a dimension of the data, and a set of attribute values, which define what attribute values are allowed to match that node. We refer to such nodes as *constrained*. As a convenient shorthand, we refer to nodes that can match any attribute value for that dimension as *unconstrained*, because they are not constrained to match a specific subset of

Table 3. Interaction scheme for adding, removing, and viewing query graph components.

Required Function	Action	Command
Add node (unconstrained)	Creates an unconstrained node in the query graph	**Select** node type from node type list. **Press *N*** in the query graph view.
Add node (constrained)	Creates a constrained node to the query graph	**Select** desired attribute values in view displaying attribute values for the desired node type. **Press *N*** in the view displaying the attribute values.
View node constraints	Displays attribute values that a node is allowed to match	**Select** constrained node in the query graph. Browse constraints in the list of node constraints adjacent the query graph view.
Remove node	Deletes a node from the query graph	**Select** desired node. **Press *DELETE*** in the query graph view.
Add conjunctive relationship between nodes	Creates a hyperedge between nodes	**Select** a node in the query graph. **Press *P*** in the graph view. **De-select** the node that is now in the hyperedge. **Select** the hyperedge. For each additional node, **select** the node, **press *P***, and **de-select** the node.
Remove conjunctive relationship	Deletes a hyperedge from the graph	**Select** the hyperedge. **Press *DELETE*** in the query graph view.

attribute values. Hyperedges are non-empty sets of nodes which define conjunctive relationships between the nodes they contain.

Hypergraph queries represent conditional statements. A node matches a data item if the data item's attribute value for the dimension specified by the node type exists in the set of attribute values for that node. Put another way, if any of the attribute values in that node's set of allowable matches is present in the data item's attribute values for that dimension, then the data item matches that node. A hyperedge matches a data item if all of the nodes within that hyperedge match the data item. A hypergraph query matches a data item if any of the hyperedges matches the data item.

Described another way, nodes in a hypergraph are statements describing intra-dimensional disjunctive relationships; i.e., logical *OR* relationships between attribute values. Hyperedges are statements describing inter-node conjunctive relationships; i.e., logical *AND* relationships between the statements described by the nodes they contain. Hypergraphs are statements describing inter-hyperedge disjunctive relationships: logical *OR* relationships between the statements described by the hyperedges in the graph. Figure 4 shows example hypergraphs representing queries on a book catalogue.

Formally, we define a query graph $q = (N, H, T, V, f)$ in which

- N is a set of nodes
- H is a set of hyperedges which are non-empty subsets of N
- T is a set of attribute types
- V is a set of attribute values
- f is a surjective function $f : T \mapsto V$.

A node $n \in N$ is a pair (t, U), where $t \in T$ defines the type of the node, and $U \subset f(t)$ defines the set of attribute values that are considered a match to this node. A hyperedge $h \in H$, $h \subset N$ defines a conjunct relationship between the nodes in h.

To compare a query graph $q = (N, H, T, V, f)$ to a data item d, where d is a surjective function mapping $d : T \mapsto f(t)$, we can translate the query into a conditional statement using the following rules:

- For a node $n = (t, U)$, $n \in N$, n matches $d \iff U \cap d(t)$
- For a hyperedge $h \in H$, h matches $d \iff \forall n \in h$, d matches n
- Query q matches $d \iff \forall h \in H$, d matches h

As convenient shorthand, we refer to any node $n \in N$ where $U = f(t)$ as *unconstrained*. Nodes where $U \subset f(t)$ are referred to as *constrained*.

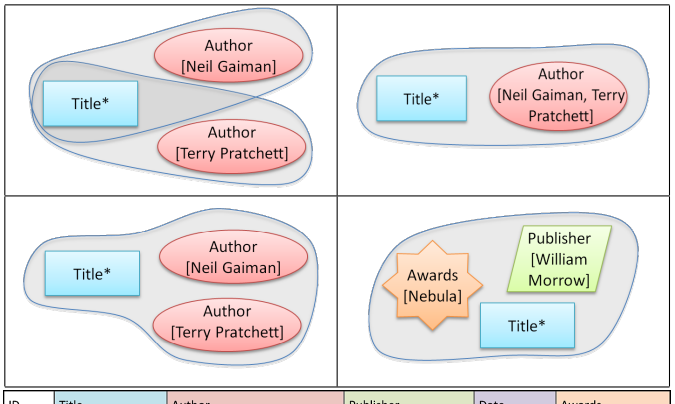
Query graphs are constructed interactively using a combination of item selection in per-dimension views and keyboard commands. There are a wide variety of interaction schemes that could have been implemented; keyboard commands were chosen based on the first letter of the word describing the concept; see Table 3.

7.2 Matching Queries

The query matching algorithm that we implemented is a scan over a mapped table, as opposed to one of the graph matching algorithms discussed in Section 6.3. There are two phases: the first sets up the mappings on the data table, and occurs only once. The second is the hypergraph evaluation, which is a linear scan of the table and occurs every time a query is evaluated.

In the mapping phase, mappings are created for each queryable dimension to map the primary key of the main table to the set of attribute values present for that dimension for that record. The mappings are used to quickly access all attribute values in a particular dimension for a given record.

In the evaluation phase, the hypergraph query is iteratively evaluated against each record in the table. Each hyperedge in the query is evaluated by comparing the attribute values in the data record to the acceptable attribute values for each node in the hyperedge. If all of the nodes in the hyperedge match the data record, that data record is a match for the hyperedge.



ID	Title	Author	Publisher	Date	Awards
1	American Gods	Neil Gaiman	William Morrow	09/02/2003	Hugo, Nebula
2	Neverwhere	Neil Gaiman	William Morrow	09/02/2003	
3	Good Omens	Neil Gaiman, Terry Pratchett	Guild	5/1/1990	
4	Neuromancer	William Gibson	Ace Books	7/1/1984	Hugo, Nebula
5	Going Postal	Terry Pratchett	HarperTorch	9/27/2005	Nebula
6	Xenocide	Orson Scott Card	Tor Books	8/15/1992	Hugo

Fig. 4. Example hypergraph queries on a catalogue of books. The top two queries are alternative ways to express *"Which books were written by Neil Gaiman OR Terry Pratchett?"*, resulting in answers *American Gods*, *Neverwhere*, *Good Omens*, and *Going Postal*. The bottom left query expresses *"Which books were written by Neil Gaiman AND Terry Pratchett?"*, producing *Good Omens*. The bottom right query expresses *"Which books published by William Morrow won Nebula awards?"*, giving the answer *American Gods*.

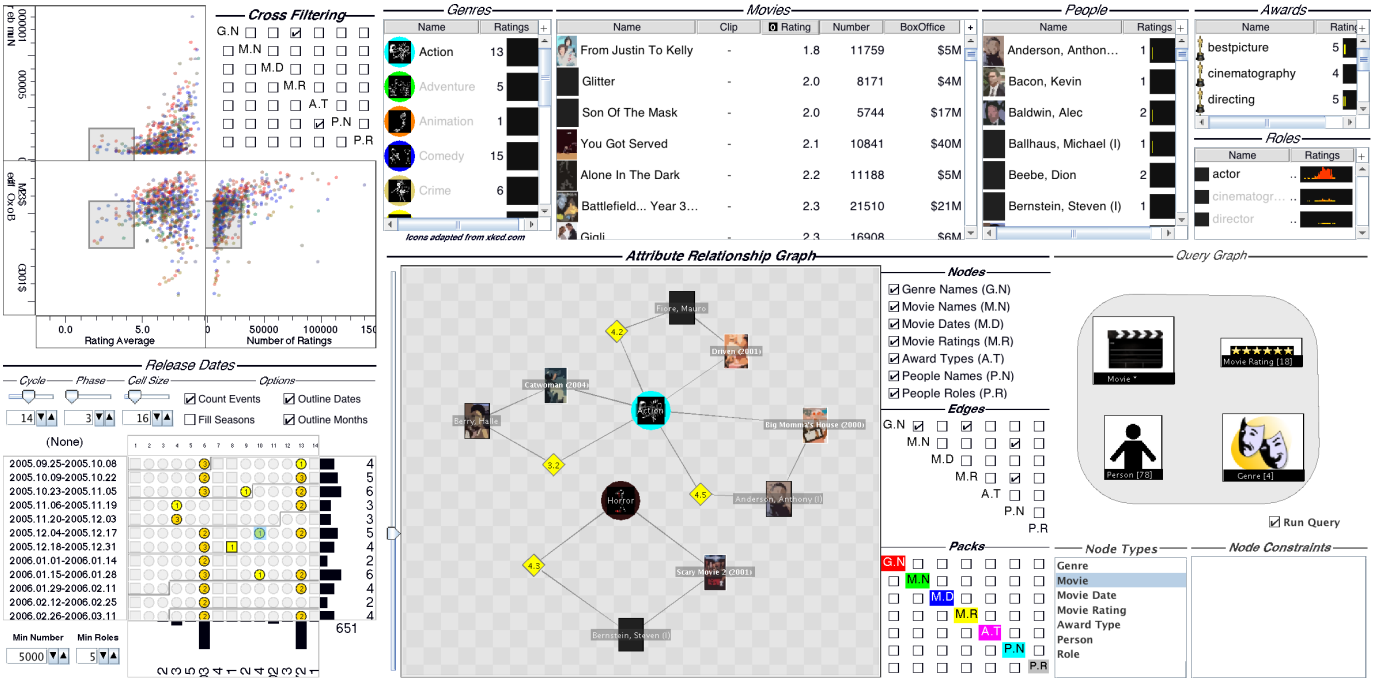


Fig. 5. Cinegraph extended, displaying actors involved with movies that won awards who were also involved in movies with a low rating.

The worst case time complexity of our implementation is $\mathcal{O}(m(2^n - 1))$, where m is the size of the table and n is the number of nodes in the query hypergraph. However, the average case is significantly better. The worst case is for a fully connected hypergraph, in which hyperedges exist between every possible subset of nodes. According to the definition of the language, this is equivalent to a hypergraph with a single hyperedge containing all of the nodes. Provided that the user understands the querying system and acts reasonably, the worst case is highly unlikely to occur. Furthermore, while the table size m may be quite large, n tends to be small. Simple queries involve two to four nodes and two hyperedges, and even complex queries are unlikely to involve more than ten nodes and two or three hyperedges. This keeps the execution time manageable.

7.3 Interactive Visual Tools

A second example tool, shown for two different hypergraph queries in Figures 5–6, is an extension of the Cinegraph visualization [27]. Cinegraph visualizes a subset of the Internet Movie Database [12] (IMDb) using a combination of the cross-filtering and attribute relationship graph techniques. The IMDb is a website that allows users to find information about movies. Cinegraph visualizes a variety of movie information, such as the names of the people involved in a film, the jobs those people held on set, the date the film entered theaters, the genres of the film, any awards the film won, and the rating of the film.

The extended Cinegraph visualization, like Candid, is composed of multiple coordinated views, laid out as shown in Figure 5. A wrapping calendar shows the dates the films were released. A three-dimensional scatter plot matrix shows user ratings, number of ratings, and box office numbers plotted against each other. Users can select ranges of these three dimensions using rectangular lenses in the plots. A series of tables are populated with movie, people, role, and award information. These various views are cross-filtered by toggling checkboxes in a cross-filtering matrix at top left. Query graphs are constructed in the bottom right hand corner; the results of the query are shown in the attribute relationship graph at bottom center.

8 Discussion

In this section, we provide analysis of the hypergraph query system with respect to the goals outlined in Section 6. First, we demonstrate the expressiveness of the query graph system with examples from the classes of queries that it can express. Then, we describe the class of questions that the system cannot express without the aid of cross-filtering. Finally, we discuss some of the potential usability issues.

8.1 Questions Hypergraph Queries Can Express

The hypergraph query system, used with an attribute relationship graph, allows for exploration of n-ary, conjunctive and disjunctive, many-to-many, inter-dimensional and intra-dimensional relationships. While cross-filtering on an attribute relationship graph can express all n-ary, conjunctive and disjunctive, many-to-many, inter-dimensional relationships, in practice it is difficult to conceptualize and recall the interactions necessary to produce queries about higher-order relationships. To construct an n-ary conjunctive relationship using cross-filtering, every dimension in the query must be filtered on all previous dimensions. This requires every query to be carefully planned, as the query must be constructed in a particular order. The hypergraph query

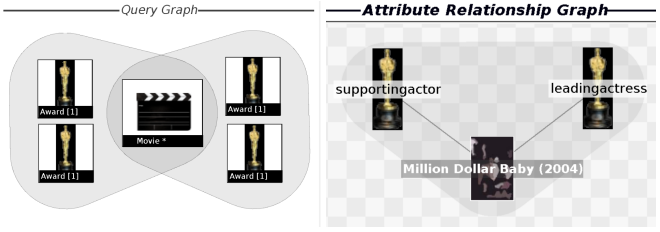


Fig. 6. Awards for Best Actress and Best Supporting Actor or for Best Actor and Best Supporting Actress. A disjunctive query between two ternary intra-dimensional conjunctive sub-queries expresses the question *"Which movies won awards either for best actor and best supporting actress or for best actress and best supporting actor?"* The *Award Type* nodes in the left hyperedge are constrained to Best Actor and Best Supporting Actress. The *Award Type* nodes in the right hyperedge are constrained to Best Actress and Best Supporting Actor. The *Movie* node is unconstrained and can match any value. The only matching film in the visualized data set is *Million Dollar Baby*, for Best Actress and Best Supporting Actor.

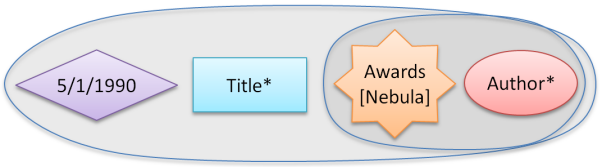


Fig. 7. Query with nested hyperedges constructed for data in Figure 4.

system simplifies the process of higher-order query development by consolidating the query expression into a single view and reducing the number of interactions necessary to build a query. Higher-order conjunctive queries in the hypergraph system may be constructed in any order, reducing mental load for the analyst.

8.2 Questions Hypergraph Queries Cannot Express

While the hypergraph query system can flexibly express n-dimensional inter and intra-dimensional conjunctive queries as well as disjunctive relationships between higher-order conjunctive sub-queries, it is unable to express queries in which a conjunctive relationship restricts which data items match a node contained in the conjunctive relationship. Consider the query in Figure 7. We would like this query to mean *"Which books written by authors who won Nebula awards were published on 5/1/1990?"* The matching record for this query would be *Good Omens*, as it is the only book published on 5/1/1990 that also has an author (Neil Gaiman) who won a Nebula award. What this query actually means, however, is *"Which books that won Nebula awards were published on 5/1/1990, and who wrote them?"* Fortunately, this kind of query can be answered using cross-filtering in combination with the hypergraph query system.

Cross-filtering was used to support the query-graph building process for the query shown in Figure 5. The questions being asked are *"Which actors, who have been involved with a movie that won an award, have also been in an Action, Comedy, Drama, Sci-Fi, Thriller, or Horror movie with a low rating? What was the movie with a low rating?"* These questions cannot be directly expressed using the hypergraph query system. While the answer could possibly be found by posing a less specific question using the hypergraph query system, the high cardinality of data values in the movie and people dimensions means that it would be impractically difficult to find the answer on the resulting attribute relationship graph. Cross-filtering allows us to sidestep this problem by first filtering the people dimension on selected awards, leaving the people dimension populated only with those people involved with films that won awards, and then restricting the people nodes to only those individuals.

Cross-filtering is not the only way to achieve this effect, however. If sets of nodes can be selected in the attribute relationship graph and then added to query nodes as constraints, the pre-filtering capability can be provided by iterative querying. Instead of selecting the awards of interest and filtering People on Awards, we would create a query graph with a single hyperedge containing an unconstrained Person node and an Award Type node constrained to the awards of interest. The results for that query would be the set of people who had been involved with a movie that had won an award, precisely the same subset as we get from filtering People on Awards. We could then select that subset in the graph and use it to constrain a Person node, before continuing to build the rest of the query.

This technique is particularly attractive because it facilitates a more iterative form of data exploration. A drawback of the hypergraph query system is that it works better if one has a well-formed question to pose. Encouraging iterative construction of query graphs would make the system more accessible to users who may not have a good idea of what is in the data.

9 Future Work

We have identified three opportunities to extend the hypergraph querying technique. One such extension would be a complement operator for nodes and hyperedges in the query graph. Complementing a node in the query graph would restrict the matches for that node to any attribute value except the attribute values specified by that node's set. Complementing a hyperedge in the query graph would restrict the matches for that hyperedge to any record that does not conform to the relationship specified by that hyperedge. This extension would allow query graphs to more closely conform to the structure of questions in natural language.

A second extension would be the ability to designate nodes in the query graph as "the same as" another node. Currently, a user has no way to ask questions like *"Which letters were sent from and to the same country?"* unless they constrain the nodes that need to be the same to the same single value. This would be particularly helpful when the commonalities are of interest, but the specifics of the commonalities are unknown. Related to this is the ability to merge dimensions of data with intersecting sets of attribute values. For instance, in Candid, the Author Occupation and Recipient Occupation are separate dimensions with overlapping sets of attribute values. In instances in which we do not care who the occupation was held by, only that it was associated with another dimension, it may be helpful to specify a unified "Occupation" dimension that subsumes both Author Occupation and Recipient Occupation.

A third extension would be the ability to define nested hyperedges to express the set of questions that the hypergraph querying system is unable to express without the help of cross-filtering. However, it is unclear whether this extension would be more beneficial than a less visually complex solution of allowing subset selection within the attribute relationship graph for constraining nodes in the query graph.

Finally, a participant-based evaluation of the utility and usability of the hypergraph query system is needed. Humanities scholars, librarians, and intelligence analysts are likely user groups; however, features like Facebook's GraphSearch [30] suggest growing interest in providing tools for users to ask complex questions about their social networks. An evaluation of the technique should therefore include participants of varying levels of expertise from a variety of domains. The hypergraph query technique should be evaluated on two fronts: its usability (in comparison to other techniques, where applicable), and its value as an analytic tool. The former can be assessed by observing participants perform a variety of query specification tasks by building simple and complex query graphs. The latter may be evaluated longitudinally by releasing visual analysis tools incorporating the hypergraph query system to participants in a variety of domains, and interviewing the participants after a period of tool use in their research.

10 Conclusion

As our ability to collect and store data increases, so will our need for techniques to analyze it. Multidimensional data analysis requires tools for exploring the relationships between dimensions that are flexible and powerful. In this paper, we provide a detailed analysis of the space of questions expressible using cross-filtering and cross-filtering on an attribute relationship graph, as well as a discussion of the design considerations for developing a visual graph query technique to complement and expand the expressiveness of cross-filtering on an attribute relationship graph. We present a specification of a hypergraph-based query technique that allows users to construct queries on multidimensional data by combining nodes and hyperedges. Using this hypergraph representation, users can represent n-ary conjunctive inter- and intra-dimensional queries involving many-to-many relationships in a way that promises to reduce cognitive load and allow for more complex explorations of richly heterogeneous multidimensional data.

Acknowledgments

The authors wish to thank our Digging Into Data collaborators Dan Edelstein, Paula Findlen, and Nicole Coleman of the Stanford Humanities Center. This work was supported in part by National Science Foundation Award #1036331.

REFERENCES

[1] V. Batagelj and A. Mrvar. Pajek-program for large network analysis. *Connections*, 21(2):47–57, 1998.

[2] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J.-D. Fekete. GraphDice: A system for exploring multivariate social networks. In *Computer Graphics Forum*, volume 29, pages 863–872. Wiley Online Library, 2010.

[3] H. Blau, N. Immerman, and D. Jensen. A visual query language for relational knowledge discovery. Technical report, University of Massachusetts, Amherst, Amherst, MA, 2001.

[4] U. Brandes and D. Wagner. Analysis and visualization of social networks. In *Graph drawing software*, pages 321–340. Springer, 2004.

[5] D. H. Chau, C. Faloutsos, H. Tong, J. I. Hong, B. Gallagher, and T. Eliassi-Rad. GRAPHITE: A visual query system for large graphs. In *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 963–966. IEEE, December 2008.

[6] M. P. Consens and A. O. Mendelzon. GraphLog: A visual formalism for real life recursion. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 404–416, Nashville, Tennessee, 1990. ACM.

[7] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactopms on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372, 2004.

[8] H. Ehrig. Introduction to the algebraic theory of graph grammars (a survey). In V. Claus, H. Ehrig, and G. Rozenberg, editors, *Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69. Springer Berlin / Heidelberg, 1979.

[9] R. H. Güting. GraphDB: Modeling and querying graphs in databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 297–308, Santiago, Chile, 1994. Morgan Kaufmann Publishers Inc.

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

[11] N. Henry and J.-D. Fekete. MatrixExplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):677–684, September/October 2006.

[12] Internet Movie Database (IMDb). Internet movie database (IMDb). http://www.imdb.com/, 2012.

[13] Z. Liu, S. B. Navathe, and J. T. Stasko. Network-based visual analysis of tabular data. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 39–48, Providence, RI, October 2011. IEEE.

[14] B. D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[15] J. O'Madadhain, D. Fisher, S. White, and Y. Boey. The JUNG (Java universal network/graph) framework. *University of California, Irvine, California*, 2003.

[16] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *Visualization and Computer Graphics, IEEE Transactions on*, 12(6):1427–1439, 2006.

[17] L. Sheng, Z. M. Özsoyoğlu, and G. Özsoyoğlu. A graph query language and its query processing. In *Proceedings of the 15th International Conference on Data Engineering*, pages 572–581, March 1999.

[18] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, November 1994.

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

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

[21] T. J. Teorey, D. Yang, and J. P. Fry. A logical design methodology for relational databases using the extended entity-relationship model. *Computing Surveys*, 18(2):197–222, June 1986.

[22] H. Tong, B. Gallagher, C. Faloutsos, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *Proceedings of Knowledge Discovery in Databases (KDD)*, pages 737–746, San Jose, CA, August 2007. ACM.

[23] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42, Jan. 1976.

[24] University of Oxford. Electronic Enlightenment. http://www.e-enlightenment.com/, 2012.

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

[26] C. Weaver. Building highly-coordinated visualizations in Improvise. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 159–166, Austin, TX, October 2004. IEEE Computer Society.

[27] C. Weaver. InfoVis 2007 contest entry: Cinegraph. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis) (Compendium)*, Sacramento, CA, October 2007. IEEE.

[28] C. Weaver. Cross-filtered views for multidimensional visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):192–204, March-April 2010.

[29] C. Weaver. Multidimensional data dissection using attribute relationship graphs. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 75–82, Salt Lake City, UT, October 2010. IEEE.

[30] M. Zuckerberg. Building graph search. *Facebook*, 2013.