# Methods and Tools for Visual Analytics

by

Hao Zhou

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Statistics
(Statistics)
in The University of Michigan
2011

*To my parents*

# ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Professor George Michailidis. He has unstintingly shared his fantastic excitement about, and knowledge of, statistical methods and visual analytics. He has introduced me to these beautiful subjects. In addition, he has kept me motivated throughout my graduate school years.

I have received generous support from Professor Hosagrahar V. Jagadish, who also guided me to work on the visual analytic algebra project. Discussions with him and his student, Anna A. Shaverdian, have substantially deepened my understanding of the system design. All of these are crucial in shaping my thesis.

# TABLE OF CONTENTS

# LIST OF FIGURES

# ABSTRACT

Methods and Tools for Visual Analytics

by

Hao Zhou

Chair: George Michailidis

Technological advances have led to a proliferation of data characterized by a complex structure; namely, high-dimensional attribute information complemented by relationships between the objects or even the attributes.

Classical data mining techniques usually explore the attribute space, while network analytic techniques focus on the relationships, usually expressed in the form of a graph. However, visualization techniques offer the possibility to gain useful insight through appropriate graphical displays coupled with data mining and network analytic techniques.

In this thesis, we study various topics of the visual analytic process. Specifically, in chapter 2, we propose a visual analytic algebra geared towards attributed graphs. The algebra defines a universal language for graph data manipulations during the visual analytic process and allows documentation and reproducibility. In chapter 3, we extend the algebra framework to address the uncertain querying problem. The algebra's operators are illustrated on a number of synthetic and real data sets, implemented in an existing visualization system (Cytoscape) and validated through a small user study.

In chapter 4, we introduce a dimension reduction technique that through a regularization framework incorporates network information either on the objects or the attributes. The technique is illustrated on a number of real world applications.

Finally, in the last part of the thesis, we present a multi-task generalized linear model that improves the learning of a single task (problem) by utilizing information from connected/similar tasks through a shared representation. We present an algorithm for estimating the parameters of the problem efficiently and illustrate it on a movie ratings data set.

# CHAPTER I

# Introduction

As technology advances daily, demands for processing complex high dimensional graphical data become increasingly needed in various fields. For example, in the study of social computing network, network structure contains communication information among user, while user logs provide addition characteristic information on the individual level. Moreover, in many biology studies, researchers collect expression levels for genes. The tested genes are often functionally related, while patients may be blood-related in the study of inherited diseases.

In recent years, many studies have shown that a mixture of human and machine intelligence can often be much more effective than either on alone. Since visualization is the most intuitive and direct way of learning for data, visual analytic, which is the science of analytic reasoning through visual interaction, has received a lot of attention as evidenced by large number of tools and algorithms developed for this purpose.

In order to use these tools and algorithms, one often needs to go through a growing body of literature on visual analytic systems. In addition, it is difficult to find precise documentation for how a visualization was created to display results of a given study; therefore, it creates limitation on replicating others' work. Furthermore, many visual analytic tools are centered on a particular type of application, such as biological data, then there is no clear translation technique to replicate the tool's benefits and function for other applications. The underlying cause for all these problems is the lack of a systematic method to design graph tools. There is no universal language that defines the basic graph data manipulation actions in visual analytics.

In the second chapter of the thesis, we present a visual analytic graph algebraic method to solve those problems. The visual analytic graph algebra is part of a framework with the following components:(1) a formalized graph model, (2) an expressive predicate language, and (3) an algebra with associated operators and functions. The purpose of the operators is to manipulate the raw graph data during a visual analytic process. The selection operator zooms into a region of interest. The aggregation operator manipulates the data resolution with super nodes.The labeling function tracks notable information collected during an analysis, which is useful during lengthy analysis. And finally, the Visual function incorporates the interactive aspect of visual analytics in a work flow, which makes a visualization creation stage as flexible as the data manipulation stage. The benefits of an algebra are now we can systematically replicate,compare, and assess graph visual analytics. Thus, a visual analytic algebra facilitates the production of graph information analysis. To show the practical usage of the algebra, we incorporate it into an existing graph analysis tool, Cytoscape. Moreover, we demonstrate the uncertainty model on a synthetical data analysis. Finally, we conduct a user study to examine its effectiveness in visual analytic work reproducibility.

In the third chapter, we propose an uncertain model based on the existing algebraic framework to handle uncertainty in the querying workflow. It is constructed from a user defined probability set based on the given uncertain query, where the probability describes how well the given outputs capture the user's target structure. Instead of only dealing a single uncertain query analytic step, it provides the user a systematic way of ranking all outputs for a multi-step visual analytic process.

Before applying any sophisticated statistical learning method to a visual analytic process, one should always visualize the data to have a basic understanding of the given information. There are many existing visualization methods, for example, Scatter plot, Principle Component Analysis and many classification methods. Scatter plot provides only 2-dimensional view of data at a time. It is often time consuming and misleading, when one tries to work with high dimensional data. Principle component analysis projects high dimensional data information into a low dimensional space by maximizing sample variance. It often give a quick summarization of the high dimensional information, however, it cannot capture any relationship among all observations by PCA's independent assumption. If the relationship among all observation can be fully captured by a group label, many existing classification method can be used for the visualization purpose. For example, linear discriminate analysis projects the high dimensional data information into a low dimensional space while preserving between group separations. When the relationships within the dataset become even more complicated, which cannot be explained by a single grouping label, existing dimension reduction methods fail to capture the entire complex data information. Some existing visual analytic systems tries to solve this problem by introducing a user interactive feature where one can visualize and work with multiple graphs at once; however, such attempt is also problematic due to the physical limitation of the screen space.

In the fourth chapter of the thesis, we introduce a dimension reduction method based on regular Singular Value Decomposition to address the problem of visualizing complex high dimensional data structures containing information among observations and multiple characteristics. Our approach is to incorporate network information by using penalty functions. We put connected observations together and let them have similar coordinates in the constructed low dimensional space. We demonstrate the performance of the network penalized singular value decomposition method on both synthetic data set and real world applications.

In the process of a visual analysis, we not only try to understand the underlying relationship within the given information, but also predicate behaviors for the future application. Existing machine learning methods are often used to solve the predication problem. Besides of observing a single high dimensional variable to describe characteristic information for each node/object in the network, we obtain multiple observations for each object, such as repeating measurements of the same object. In usual machine learning setup, we treat all information on each node as an independent task and build a leaner to understand the behavior for the single task. However, since in our case, we have the information of relationship among all nodes, the additional structure has been successfully used to improve the prediction model.

In the last chapter of the thesis, we introduce a multi-task machine learning method based on existing generalized linear model. Instead of treating each node or subject as a single independent learning task that was done by our predecessors, our contribution is constructing a learning model for each task based on all related/connected tasks. We demonstrate the performance of our model by comparing the predication error rate between the generalized linear multi-task model to other existing approaches on both synthetic dataset and real world applications. Last, we adopt bootstrapping method for the statistical inference.

CHAPTER II

# Visual Analytic Algebra

## 2.1 Introduction

Visual Analytics is the science of analytical reasoning through visual interaction. In recent years there has been growing interest in this field as people recognize that a mix of human and machine intelligence can often be much more effective than either one alone. Graphs are ubiquitous in many scientific fields: high-throughput "omic" sciences use graphs to study pathways, computer networks use graphs to analyze communications, and almost everyone is involved in the explosive growth of online social networks. Graphs are also particularly amenable to visual representation.

It is no surprise that graph visual analytics has received a lot of attention as evidenced by the large number of tools and algorithms developed for this purpose. (See Related Work Section below). Using these tools and algorithms, there is also a growing body of literature describing both visual analytics systems, as well as problems successfully addressed through visual analytics. These systems can be very informative, but usually constrain the reader in realizing their full value for several reasons, listed below.

One problem is assessing the completeness of a tool's exploration abilities. Science has not put a bound on an analyst's ability to explore graph data and produce findings. But, if the basic functions to manipulate and represent data during the analysis is enumerated, then assessing which tool is "better" in an exploration task is possible because then the functions each tool offers can be compared.

Another problem is replication, an important tenet of science. For example, if one tool is created for a biological application, there is no universal systematic way to replicate it for other types of applications.

The underlying cause for these problems is the lack of a systematic method to design graph tools. There is no common language that defines the basic graph data manipulation actions in visual analytics. Towards this goal, we present a visual analytic graph algebra to meet this void. The visual analytic graph algebra is part of a framework with the following components: (1) a formalized graph model, (2) graph data manipulation operations, and (3) an expressive predicate language.

The purpose of the operators is to manipulate the raw graph data before applying a visualization scheme. The selection operator zooms into a region of interest. The aggregation operator manipulates the data resolution with supernodes. The labeling function tracks information, which is useful during lengthy analysis. And finally, the visual function incorporates the interactive aspect of visual analytics in a work flow, which makes visualization as flexible as the data manipulation stage.

The benefits of an algebra are that one can systematically replicate, compare, and assess different graph visual analytics. Thus, a visual analytic algebra facilitates the production of graph information analysis. Based on the existing algebra, we also introduce an uncertainty model to handle queries. We propose a systematical ranking of outputs for an analytic process.

Further to make the algebra suitable for application, we present an implementation in Cytoscape (*Collaboration* (2006)), a graph visualization tool. This implementation is one example of how to implement the algebra. We will test the algebra's abilities to replicate and share graph analysis through a user study.

The remainder of the chapter is structured as follows: We begin with a related work section, 2.2. Then we describe the graph model in 2.3. Then we present the predicate language and operators from 2.4 through 2.5. Last, we demonstrate an implementation of the algebra in Section 2.6 and present the user study in Section 2.7.

## 2.2 Related Work

Our work spans related areas in graph analysis and visualization. We briefly survey selected recent works and relate them to the visual analytic graph algebra.

### 2.2.1 Graph Analysis

Several tools and methods have been created for analyzing graph data sets. One method is to capture the local topology of a graph as a signature to aid exploration of complex networks in *Wong et al.* (2006). Other works use semantic and structural abstractions to analyze social networks as in *Henry et al.* (2007). In regards to interactive graph analysis, *Tulip* (2004) presents ways to select subgraphs and manipulate the graph to find interesting relationships. There are also systems, such as in *Auber et al.* (2003), that investigate different layouts to display complex networks, their nodes, edges, and attributes effectively for analysis. A key point to note is that while there are several visual analytic tools for graph analysis, there is not a formal foundation to develop these systems.

Often to make sense of raw data, aggregating nodes into supernodes and edges helps visualize new patterns as stated in *Holten* (2006). We provide an aggregation operator which maintains the hierarchy of the aggregated nodes through descriptive attributes and edges to the original structure. The user decides which visualization scheme to apply to this transformed data. At the data level, no information is lost to an aggregate supernode.

### 2.2.2 Graph Visualization Tools

There are several noteworthy visualization tools for data analysis. For example, Cytoscape (*Collaboration* (2006)) is primarily designed for exploring biological networks. With a collection of user-community created plug-in tools, Cytoscape is able to do graph querying on basic attribute values and creation of new graphs based on selections. Other popular graphical analysis tools include Pajek (*Batagelj and Mrvar* (2009)). Pajek provides several predefined metrics for nodes and edges. This functionality is similar to our composition functions. Guess has a built in query language into its graph visualization tool. Tableau by *Hanrahan* (2006) uses a structured query language for data visualization of relational databases, cubes, and spreadsheets. ZAME (*Elmqvist et al.* (2008a)) is a visualization tool for exploring graphs at a scale of millions of nodes and edges. As pointed out above, our algebra incorporates the functionality necessary to manipulate, reduce, and analyze the graph. Such operations can directly communicate with other visualization tools through visualization operators to facilitate analysis of networks.

### 2.2.3 Visual Analytic Frameworks

Recently there has been attention to the incomplete visual taxonomy (*Laramee and Kosara* (2007)) available to visual analysts. Also, there is a challenge in the visual analytic community to develop frameworks and languages for visual analytic systems, such as *Heer et al.* (2005), *Schneidewind and Ziegler* (2006), as well as building statistical analysis tools for visualization systems in *Thomas and Cook* (2005). A number of approaches have been introduced to address the issue of coordinated multiple visualizations of data sets (*Stasko et al.* (2007), *Weaver* (2008)). Coordinated view systems are independent to the visual analytic algebra. Our aim is to provide a formal visual analytic model to manipulate graph representations combined with interactive visualization techniques. Since the scope of visual analytics includes raw data manipulation, there needs to be a bridge between the data manipulation and visualization steps.

## 2.3 Model

In this section we describe the attributed graph structure, the computed and intrinsic attributes. We also define the compositional functions, which are used to create computed attributes.

### 2.3.1 Structure

The central object of interest is an attributed graph, defined as $D = [G, X]$, where $G = (V, E)$ denotes a graph with vertex set $V \in \mathcal{V}$, edge set $E$, and attribute structure $X$. Each node $v \in V$ has a unique *id* attribute with a value assigned by an *id* label function $\lambda$. For example,$\lambda(V) = i$ is a node whose $id = i$. Since the network can be multi-edge, the edge set can be described with $E \subseteq (V \times V, \mathbb{N})$. Each edge can be referenced uniquely by its incident nodes and the identification value (e.g. $e_{i,j,n} = (v_i, v_j, n)$). In the case of a simple network, we use $e_{i,j}$.

The attribute structure $X$ associated with graph $G$ has three components: $X = [X_V,\ X_E,\ X_G]$, where $X_V$ contains node attributes (e.g. in- and out-degree), $X_E$ edge attributes (e.g. edge betweenness, direction) and $X_G$ graph attributes (e.g. diameter).

### 2.3.2 Attributes

Each attribute has a label, type, and a value. The label denotes the name of the attribute. For example: *color*, *last name*, and *age* are example labels with possible values *green*, *Smith*, and 42, respectively.

The attributes can be either an intrinsic or a computed type. Intrinsic attributes are independent features that stay the same even if other features or the graph topology changes. Intrinsic node attribute $i$ can take value in $\mathbb{R}, \mathbb{N}$ or any predefined value sets. The same applies to edge and graph intrinsic attributes.

Computed attributes are features that change as a graph structure changes. We give an example network to illustrate the structure of an attributed graph.

**Example II.1.** (Cell Phone Network) In the dataset from the VAST 2008 Challenge in *Grinstein et al.* (2008), a node represents a cell phone and an edge represents a call between two phones. This is a multi-edge directed network. The attributed graph for this data set is: $D_{phones} = \{G = (V, E),\ X = (X_V = X_{phone\_id},\ X_E = (X_{date},\ X_{duration},\ X_{tower},\ X_{direction\_of\_call}),\ X_G = ())\}$.

The attributes in $X$ are all intrinsic attributes describing features of their respective elements. The *phone_id* is the cell phone's unique identification number. The edges have several attributes including the date, duration, cell phone tower, and the direction in which the call was made. The direction attribute represents the directed nature of this graph.

### 2.3.3 Composition Functions

To obtain computed attributes there exists a collection of composition functions $\mathcal{F}$. Since there is a variety of user-defined composition functions, we focus only on their core requirements. A composition function must combine attributes from an input set and generate uniquely named computed attributes output set. If a function satisfies these requirements, we consider it in the class of composition functions.

For any $f \in \mathcal{F}$, we can express it as $f : \mathcal{Q}^{|*|} \times \cdots \times \mathcal{Q}^{|*|} \to \mathcal{P}^{|*|}$ with $* \in \{V, E, G\}$ where $\mathcal{Q}$ can take a value in $\mathbb{R}, \mathbb{N}$ or intrinsic attribute space $\mathcal{S}$ and $\mathcal{P}$ can take a value in $\mathbb{R}, \mathbb{N}$, or any predefined value sets.

Without loss of generality, we list some composition functions.

- 0 or 1 indicator function is a composition function with $\mathcal{Q} = \mathcal{S}$ and $\mathcal{P} \in \{0, 1\}$.

- Numerical aggregation functions, such as sum, average, and min/max, over a collection of attributes are composition functions.

- A function of composition functions is also a composition function; therefore, the class of composition functions is closed.

The following example shows how to construct a composition function to produce degree as listed in *Costa et al.* (2007), which is a node computed attribute.

**Example II.2.** (Counting Function) Given a simple edge network, a possible set of composition functions to compute the degree of a node, $v_i$, given its neighbor list $\mathcal{L}_{v_i}$ is
$\mathrm{COUNT}(\mathcal{L}_{v_i}) = \sum_{v_j \in V} 1_{\{v_j \in \mathcal{L}_{v_i}\}}$ summation over a list of indicator functions.
The neighbor list can be constructed based on graph information $G = (V, E)$ where $\mathcal{L}_{v_i} = \cup_j \{v_j | e_{i,j} \in E\} \bigcup \cup_k \{v_k | e_{k,i} \in E\}$. $\square$

## 2.4 Predicate Language

A predicate is essentially a query description. It allows the analyst to specify the structural and attribute conditions on the graph object she would like to manipulate. The answer to a predicate is a *witness*. To find a witness in the graph, we use a graph matching function, $\gamma$. These components of the predicate language are the basis for graph manipulation and will be used as sub-functions and inputs to the graph algebra operators.

### 2.4.1 Predicate

A necessary function in any analytical graph tool is the ability for an analyst to express the graph structure she would like to further examine or manipulate. A predicate is such structure; specifically, it is a graph structure with conditions on attributes.

Formally, a predicate tuple is $p = (V, E, \mathbb{X}_V, \mathbb{X}_E, \mathbb{X}_G, !E)$. The $\mathbb{X}_V, \mathbb{X}_E$, and $\mathbb{X}_G$ terms in the predicate are condition lists on the attribute values in $X_V, X_E, X_G$, respectively. The form for a condition in $\mathbb{X}_V$ is $X_{v.a.i}$ *op value* where $X_{v.a.i}$ is the attribute $a$ for node $i \in V$, and *op* is any relational operator. Similarly the form for a condition in $\mathbb{X}_E$ and $\mathbb{X}_G$ is written as $X_{e.b,j}$ *op value* and $X_{g.c}$ *op value*, respectively, where $j \in E$, $b$ is an edge attribute, and $c$ is a graph attribute.

The *value* is a constant or a reference to any attribute's value, irrespective of the attribute type (node, edge, or graph). Wildcards are allowed in *value*.

The form for a node attribute reference for attribute $x$ on node $k \in V$ is $X_{v.x.k}$. Note that $k$ is any node in $V$, $k \neq i$, the node on which we place the condition. An edge attribute reference for attribute $y$ on edge $j \in E$ is written as $X_{e.y.j}$. Finally, a graph attribute reference for attribute $z$ is written as $X_{g.z}$. The following is an example predicate showing attribute conditions referencing other nodes.

**Example II.3.** (Cross-Referencing) To express the predicate shown in Figure 2.1 where a node with *Phone_id* $=$ 200 and *degree* $=$ 40 has two neighbors. The neighbors' degree is described such that one has a higher degree than the other. The predicate tuple is $p = (V = (v1, v2, v3),\ E = (e_{1,2}, e_{1,3}),\ \mathbb{X}_V = ((X_{v.degree.1} = 40),$ $(X_{v.Phone\_id.1} = 200),\ (X_{v.degree.3} > X_{v.degree.2})),\ \mathbb{X}_E,\ \mathbb{X}_G, !E)$. $\square$

In the tuple $!E$ is the excluded edge list. An *exclusion* on edge $e$ specifies that $e$ must not exist in the graph $G$. Furthermore, if $G$ is defined on a "universe" $G'$, then the *excluded* edge must also not exist in $G'$. What this universe should be is usually clear in context, typically an initial input graph before any selections have been applied. Where the universe is not clear, we will choose it to be the graph $G$ itself. Note that $!E$ is subset of the complement of $E$.

**Definition II.4.** (Exclusion) Given predicate $\rho = (V, E,\ \mathbb{X}_V,\ \mathbb{X}_E,\ \mathbb{X}_G, !E)$, and a graph $G = (V_G, E_G)$. If $e_{i,j} \in !E$ then $e_{i,j} \notin E_G$. Furthermore, in a closed universe $U$, $\forall S$ where $G \subseteq S = (V_S, E_S)$ then $e_{i,j} \notin E_S$. $\square$

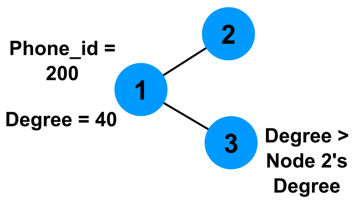


Figure 2.1: The figure shows a simple predicate corresponding to the cross-referencing example. Conditions are placed upon the attributes on the nodes. It is possible for one condition to reference the attribute on another node.

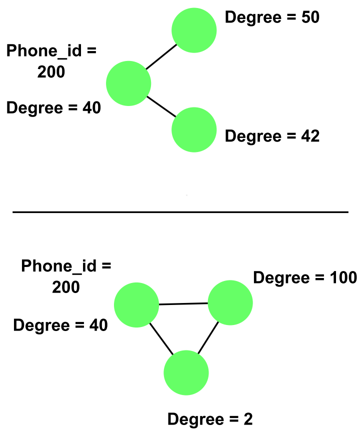


Figure 2.2: These graph structures satisfy the predicate shown in Figure 2.1. They are two possible instantiations of the predicate.

Multiple conditions on an attribute express ranges as the following example shows.

**Example II.5.** (Ranges) To express the range that node attribute $a$ must be between values $x$ and $y$, we use the following conditions: $\{(X_{v.a.i} \leq x), (X_{v.a.i} \geq y)\} \in \mathbb{X}_V$. $\square$

### 2.4.2 Witness

In the previous section we described how to define a predicate. A predicate can be thought of as the object of a question. The actual realization of a predicate in an input graph is called a *witness*.

**Definition II.6.** (Witness) Given an attributed graph $D = [G = (V, E), X = (X_V, X_E, X_G)]$ and a predicate $p = (N, M, \mathbb{X}_N, \mathbb{X}_M, \mathbb{X}_G, !M)$, $D$ is a witness to $p$ if there

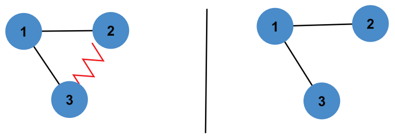


Figure 2.3: The two predicates show similar graph structures. However, the figure on the left has an excluded edge between nodes 2 and 3. Figure 2.4 shows the result of this excluded edge on possible witnesses given an input graph.

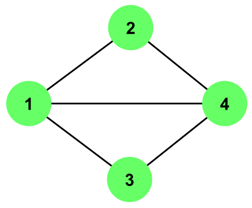


Figure 2.4: The figure above is an input graph. Given the two predicates shown in Figure 2.3 we describe the different witnesses existing in this input graph. The sets of nodes which induce a witness to the excluded edge predicate are: 1,2,3 and 2,4,3. The predicate without the excluded edge results has more witnesses: 1,2,3; 1,2,4; 1,4,3; 2,4,3; and more. Since there is an edge between node 1 and 4, this the witness 1,2,4 does not satisfy the predicate with the excluded edge.

is a bijection mapping function, $f_{D,p}$, between the vertices in $V$ and $N$ such that,

- $e_{i,j,k} = ((v_i, v_j), k) \in E$ if and only if $m_{i,j,l} = ((n_i, n_j), l) \in M$ where $v_i = f_{D,p}(n_i)$ and $v_j = f_{D,p}(n_j)$ and $k = f_{D,p}(l)$
- $\forall (X_{n.i.x}\ op\ value) \in \mathbb{X}_N$, $(X_{v.f_{D,p}(i).x}\ op\ value)$ holds in $D$, where $i \in N$ and $x$ is a node attribute label.

and an injection edge attribute assignment function, $h_{D,p}$, between the edges $E$ and $M$ such that $\forall m_{i,j,k} = ((n_i, n_j), k) \in M$

- $\forall (X_{m.j.y}\ op\ value) \in \mathbb{X}_M$, $(X_{e.h_{D,p}(j).y}\ op\ value)$ holds in $D$, where $j \in M$ and $y$ is an edge attribute label.

the conditions on $\mathbb{X}_G$, must be satisfied in $X_G$,

- $\forall(X_{g.z}\ op\ value) \in \mathbb{X}_G$, $(X_{g.z}\ op\ value)$ holds in $D$, where $z$ is a graph attribute label.

and finally, if in a closed universe $U$, $\exists\ D' = [G' = (V', E'), X' = (X_{V'}, X_{E'}, X_{G'})]$ such that $D \subseteq D'$

- if $m_{i,j,k} = ((n_i, n_j), k) \in !E$ then $e'_{i,j,w} = ((v'_i, v'_j), w) \notin E'$ where $v'_i = f_{D',p}(n_i)$ and $v'_j = f_{D',p}(n_j)$. $\square$

Both structural and attribute properties of $D$, and the predicate, $p$, must match for $D$ to be a witness of $p$. There must be a bijective mapping between the vertices and edges in the predicate structure and the witness. In other words, there must be an isomorphic matching between the node and edge identifications.

We show some examples of predicates, witnesses, and *excluded* edges.

**Example II.7.** (Witnesses) Figure 2.2 shows two example witnesses for the predicate in Figure 2.1. Both satisfy the structural and attribute conditions although these witnesses clearly are not identical graphs.

**Example II.8.** (Predicate with Excluded Edges) Figure 2.3 shows two predicates. The predicate on the left has an *excluded* edge between nodes 2 and 3. This predicate is written as $p_1 = (V = (v1, v2, v3), E = (e_{1,2}, e_{1,3}), \mathbb{X}_V, \mathbb{X}_E, \mathbb{X}_G, !E = (e_{2,3})$. Note that an edge is denoted with two indexes when the third one is not relevant. The predicate on the right is a similar predicate but does not have an *exclusion* between nodes 2 and 3. This predicate is written as $p_2 = (V = (v1, v2, v3), E = (e_{1,2}, e_{1,3}), \mathbb{X}_V, \mathbb{X}_E, \mathbb{X}_G, !E = ())$. $\square$

**Example II.9.** (Excluded Edges) The difference between predicates shown in Figure 2.3 can be seen by the supergraph in Figure 2.4. The nodes in the input graph that induce witnesses to the predicate with an excluded edge are: $\{1, 2, 3\}$ and $\{2, 4, 3\}$. There are many more sets of nodes that induce witnesses to the predicate without the excluded edge. For example, a few are: $\{1, 2, 3\}$; $\{1, 2, 4\}$; $\{1, 4, 3\}$; and $\{2, 4, 3\}$. Due to the edge between nodes 1 and 4, the subgraph induced by the set of nodes $\{1, 2, 4\}$ cannot be a witness to the predicate with the excluded edge. □

### 2.4.3 Graph Matching Function

The process of pairing a predicate to a witness is called graph matching. Graph matching is a well-studied problem in graph analysis. In terms of Visual Analytics, it is useful to see how a predicate matches a witness in an input graph. For example, a predicate might have many witnesses in an input graph. How do these witnesses compare to each other? Where are these witnesses in relation to each other in the input graph? By knowing the details of how a witness matches a predicate, an analyst can answer similar questions.

In this section we describe a graph matching function, $\gamma$. The specific graph isomorphism algorithm can be replaced with any state-of-the-art algorithm. We focus on the input and the outputs of the graph matching function.

**Definition II.10.** Given an attributed graph $D$ and a predicate $p$, graph matching $\gamma$ outputs

- A list of witnesses found in $D$. In the case of duplicate witnesses, a single arbitrary witness is returned from the duplicate set.
- A model witness $X$ which is an attributed graph instantiation of the predicate's structure and attribute conditions.
- A set of mapping lists for each witness to $X$.

Often, the same set of nodes in the graph can match at multiple positions in a structural predicate. This is trivially true when the condition applied at each node is the same. In this case, every permutation of node matches is a "new" way to satisfy

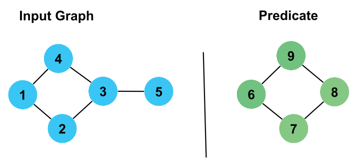


Figure 2.5: The Graph Matching function takes two inputs: a graph and a predicate. Given these inputs, the graph matching function will find witnesses that satisfy the predicate within the input graph.

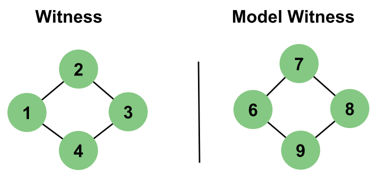


Figure 2.6: Given the inputs to the graph matching function shown in Figure 2.5, the function will return three types of output. First there exists one witness in the input graph. Second, a model witness is returned that maintains the predicate structure. In this case, the model witness is identical to the witness. But it is possible the witness contains an edge between nodes 1 and 3 and still be a witness to the predicate. The last structure returned is a mapping list for the witness to the model witness. The mapping list is useful for the analyst to see how the witness matches the predicate. The mapping list in this case is: $\{1 \rightarrow 6, 2 \rightarrow 7, 3 \rightarrow 8, 4 \rightarrow 9\}$

the given predicate. This can result in an unacceptably long list of matches. To avoid this eventuality, we consider two witnesses to be duplicate if they comprise the same set of nodes and edges, even if the matching to the structural pattern is different. The mapping lists save how each witness satisfies the predicate.

**Example II.11.** (Gamma function) Given the two inputs to the graph matching function shown in Figure 2.5, the matching function returns a single witness, a model witness, and a mapping list of the witness to the model witness. The outputs are shown in Figure 2.6.

## 2.5 Operators and Important Functions

In order to formally define the basic manipulating step in visual analytics, we introduce both selection and aggregation operators.

### 2.5.1 Selection

There are two selection methods that operate on sets of graphs: set and element selection. Set selection reduces the number of graphs in a set. Element selection reduces the nodes and edges in each graph.

#### 2.5.1.1 Set Selection

Set selection filters a set of attributed graphs that do not have at least one witness for the given predicate. For example, an analyst may start with a large collection of graphs but she is only interested in the graphs with a diameter larger than 20. Set selection is applied to focus on the interested set.

**Definition II.12.** (Set Selection) Given a collection of graphs and their attributes, $\mathcal{D}$, a set selection with predicate $\alpha$, is $\sigma^{extract}_{set,\alpha}(\mathcal{D}) = \{D \in \mathcal{D} | \text{there exists a witness in } D \text{ for } \alpha\}$. As a subroutine, the graph matching $\gamma$ function is called to determine if any witness exists for each of the attributed graphs. $\square$

**Example II.13.** Figure 2.7 shows a set of two attributed graphs. If we perform a set selection with a predicate that the average degree of the input graph be equal to 2, the result will be the first attributed graph. Each node in the result has a degree of 2. Since it is the only input graph that satisfies the predicate, it is the only graph returned.

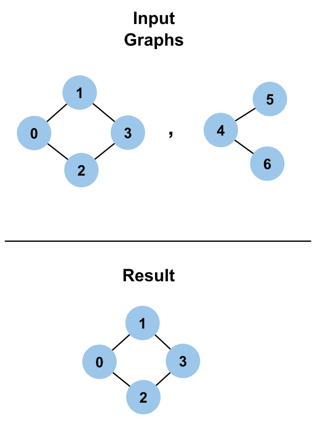


Figure 2.7: Two input attributed graphs are displayed. If the predicate to the set selection function is that the graph attribute, average degree, be equal to 2, then the result from set selection is the input graph shown on the left. The input graph shown on the right has an average degree of 1.67.

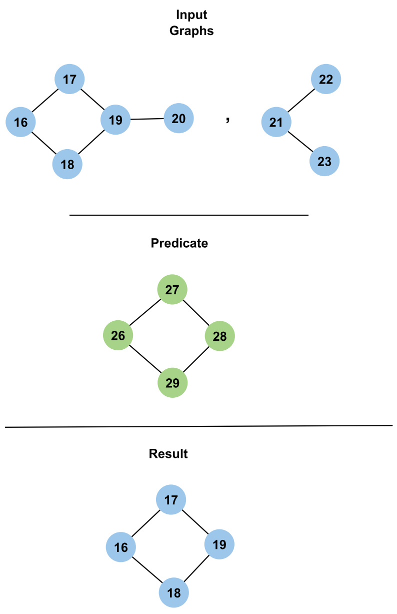


Figure 2.8: Given the two input graphs and the predicate shown above, the resulting attributed graph after an element selection call is shown. The input graph contains a witness for the predicate, namely the witness induced by the node set (16,17,18,19). The input graph on the right has no witness for the given predicate. The graph matching function is called to determine if a witness exists in an input graph.

### 2.5.1.2 Element Selection

Similar to set selection, element selection zooms into points of interest. In this case, nodes and edges within graphs are selected. For example, an analyst may have several graphs but would like to only focus on the nodes with a degree greater than 5, in other words, remove all the other nodes. She would use element selection.

Given a set of attributed graphs and a predicate, element selection creates a new attributed graph for each witness match found. Since a new attributed graph is created, the previous computed attributes may not apply to the new topology of the graph. Therefore, computed attributes are updated for the new attributed graphs.

**Definition II.14.** (Element Selection) Given a collection of attributed graphs, $\mathcal{D}$, an element selection with predicate $\alpha$, is $\sigma^{selection}_{element,\alpha}(\mathcal{D}) = \cup_i\{W_i'\}$ that satisfies

- $D_i = [G_i, X_i] \in \mathcal{D}$
- $W_i = \{W_{i.\alpha.1}, \cdots, W_{i.\alpha.k}\}$ for $k$ = number of witness of $\alpha$ on $D_i$

where each $W_{i.\alpha.j} = [G_{i.\alpha.j}, X_{i.\alpha.j}]$ is the $j^{th}$ witness of $\alpha$ on $D_i$.

The attributes in each witness, $W_{i.\alpha.j}$, are as follows:

1. All graph attributes from $D_i$ carry through to $X_{i.\alpha.j}$
2. All node and edge attributes in $D_i$ which are attributes of the nodes and edges in $W_{i.\alpha.j}$ carry through to $X_{i.\alpha.j}$
3. All computed attributes in $X_{i.\alpha.j}$ are updated $\square$

**Example II.15.** Figure 2.8 shows two input graphs and a predicate. Element selection returns all structures that match the given predicate structure. The only attributed graph returned is from the input graph on the left. No witness for the predicate exists in the input graph shown on the right.

### 2.5.2 Aggregation

We use aggregation to reduce the complexity of a graph by combining structures with similar patterns. These patterns are stated as a list of predicates.

Aggregation operates on both the set level and the element level. At the set level, aggregation unions the attributed graphs together that satisfy a predicate. The result is a new set of attributed graphs, one for each predicate. At the element level, aggregation merges the nodes or attributes together that satisfy a predicate in the list.

#### 2.5.2.1 Set Aggregation

Set aggregation performs a union over the elements of the attributed graphs that satisfy a given predicate. For each of the predicates, set aggregation checks each of the $n$ input graphs for at least one witness.

For the input graphs that satisfy the predicate, set aggregation will union their elements (graph and attributes) and produce a single output for the predicate. The aggregation performs as a union of a set of disjoint attributed graphs. The elements of the inputs are given a new unique identification in the new outputs. Algorithm for set aggregation pseudocode is listed in appendices.

**Definition II.16.** (Set Aggregation) Given a set of predicates, $\beta = \{\beta_1, \cdots, \beta_m\}$, and a set of attributed graphs, $\mathcal{D}$, set aggregation is defined as $\phi_{set.\beta}(\mathcal{D}) = \mathcal{D}' = \{D'_1, \cdots, D'_m\}$. where each $D'_i$ are the union input attributed graphs that satisfy predicate $\beta_i$.

**Example II.17.** Given a set of phone call networks, each representing a single day (with a graph attribute $X_{G,day} = i$), over a four day period, we wish to aggregate days one to two and aggregate days three to four. In this case we call $\phi_{set.\beta}(\{D_{day1}, D_{day2}, D_{day3}, D_{day4}\})$ where $\beta$ is the predicate set with one predicate on days 1-2 and the other on days 3-4. This set aggregation produces the following set $\{D_{days1-2}, D_{days3-4}\}$ $\square$

#### 2.5.2.2 Element Aggregation

Element aggregation merges the nodes and edges of an attributed graph by groups. The groups are described by a set of predicates. There are two types of groups: group by structure or group by all structures.

In group by structure, element aggregation creates a new node with a unique id for each witness found for each predicate. The new node represents a merged supernode. The attributes of the new node are aggregated values of the nodes contained in the witness. The supernode also maintains the connections to the model witness node and to the nodes in the graph which the aggregated nodes connected to before being combined into the supernode.

Group by all structure is similar to group by structure except supernode is not created for each witness found. Instead, one supernode is created to represent all the witnesses for a predicate.

**Definition II.18.** (Element Aggregation) For a given predicate list $\beta = \{\beta_1, \cdots, \beta_m\}$, an attributed graph set $\mathcal{D}$, element aggregation is defined as $\phi_{element.\beta.type}(\mathcal{D})$ where *type* is either by structure or by all structures. Element aggregation modifies the input graphs into aggregated graphs is listed in appendices.

We show the difference between aggregation by structure and by all structures with examples.

**Example II.19.** (Element aggregation by all structures) Figure 2.9 shows the effect of an element aggregation on an input graph. The predicates we pass are $\alpha = \{\alpha_1 = \{V = \{V_1\}, X_{v.color.1} = blue\}, \alpha_2 = \{V = \{V_1\}, X_{v.color.1} = purple\}\}$.

After the aggregation, nodes 1, 2, and 4 are merged together, which is $\alpha_1$'s witness group; and, nodes 0 and 3 are merged together, which is $\alpha_2$'s witness group. The

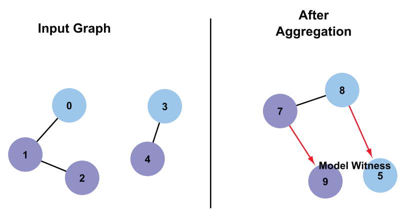


Figure 2.9: An element aggregation by all structures is performed on the input graph. The result is the blue nodes are merged into one group, and the purple nodes into a second group.

computed attributes for the new merged nodes 7 and 8 are aggregated values from their merged nodes. The attribute type determines how to aggregate the values. For example, numbers may be averaged together.

Model witness nodes 9 and 5 are created to save the predicate match structure and attribute conditions. Their purpose is to remind the analyst in the future why these nodes were aggregated. There are directed edges from the supernodes to their respective model witnesses. This added attribute and structural information differentiates a supernode from non-aggregated nodes. □

**Example II.20.** (Element aggregation by structure) In Figure 2.10 there is an input graph containing three triangular structures. An element aggregation *by* structure given the predicate shown will create three merged nodes. The two merged nodes that are connected to each other result from the connection between two of the triangular structures in the input graph. In the "after aggregation" picture, we can see the merged triangular structures within the new nodes. Figure 2.11 shows how the merged structures within the new nodes are stored. This is done by creating a model witness for the predicate. The new nodes point to the model witness structure.

### 2.5.3 Labeling function

Just observing a network can lead to many mental notes. But for every observation, it is too cumbersome to open a new branch of analysis. In terms of our algebra,

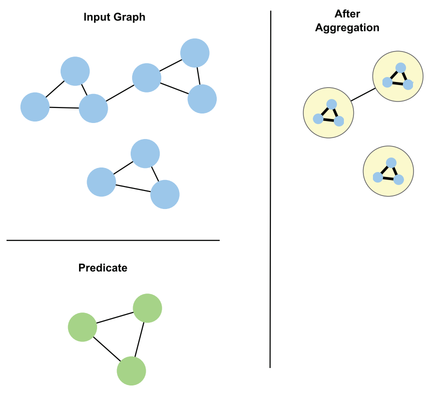


Figure 2.10: After an element aggregation by structure, the attributed graph becomes only three nodes.

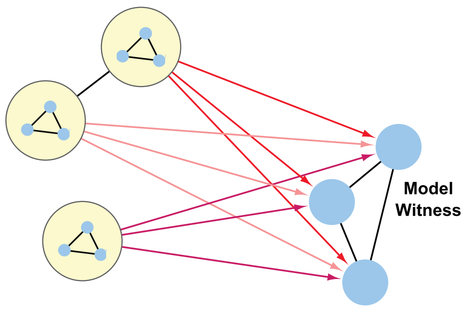


Figure 2.11: The three aggregated nodes in the merged graph of Figure 2.10 point to a model witness. The purpose of this model witness is the retain the predicate structure, the reason for the aggregation, to understand the analytical process.

we may prefer not to have to perform a selection and aggregation to create a new graph. Sometimes, simply making a note on the current graph will suffice. Then as analysis continues, these notes could be saved and easily tracked. The structure of this "mental note" on a graph leads us to our next operator: Labeling.

Labeling does not create a new graph, but it creates structured attributes based on a process similar to the element selection. The inputs to the labeling function are a set of attributed graphs and a predicate.

As a result of labeling, the attributed graphs that contain a witness are modified to include edges to a model witness (for the predicate). The model witness, similar to element aggregation, is given a new group id. This group id is used to identify it as an informational structure within the graph (not to be included in selections).

Below is a formal definition of labeling:

**Definition II.21.** (Labeling) Given a collection of $n$ graphs and their attributes $\mathcal{D}$, and predicate $\alpha = (V_\alpha, E_\alpha, X_{V_\alpha}, X_{E_\alpha}, X_{G_\alpha})$, a labeling, $\sigma^{label}_{element,\alpha}(\mathcal{D})$ is: For each attributed graph $D_i$ in $\mathcal{D}$ where there exists a witness for the predicate $\alpha$,

1. Create the model witness structure $X$ within $D_i$
2. Label $X$'s nodes and edges with a unique group id (can be user defined)
3. For each witness $W_j$ found in $D_i$ use the mapping lists to create directed edges from the nodes in $W_j$ to the nodes $X$. For the new edges created in $W_j$ to $X$ create two attributes: one to denote that it is part of one witness structure and another for the unique group id of this model witness.

□

**Example II.22.** In Figure 2.8, we show how to create new graphs based on an input set and predicate after applying the element selection operator. Sometimes an analyst

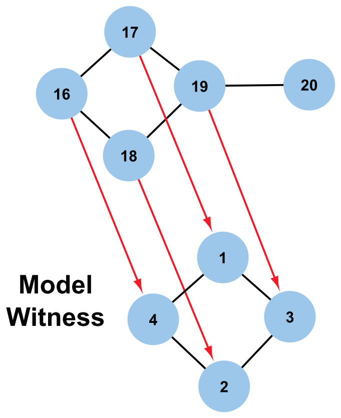


Figure 2.12: A labeling function performed on the same input graph and predicate from the element selection example, Figure 2.8

does not want to continually create more networks, rather within an exist network make notes and follow certain nodes of interest.

In Figure 2.12 we show how the labeling operator achieves this goal. Given the same input graph set and predicate from Figure 2.8, the result of the labeling operator is a model witness included into the graph (for the given predicate) and directed edges between the matched components of the input graphs. The model witness is not included in graph matching steps. □

### 2.5.4 Visualization function

Preceding sections dealt with graph data manipulation and management. In Visual Analytics, the visualization is a central step in relaying the results to the human analyst. Therefore, to complete the Graph Visual Analytic Algebra, a visual operator is needed to connect the data manipulation to the analyst. There are limitless visual encoding and rendering expressions. The challenge is to create a visual operator without limiting possible visual expressions of information.

In an effort to make this a systematic process, we break down the components of a visual function. A visual function, $\mathcal{V}$, is described by an input predicate, a static visualization output, and a visual rendering/encoding mechanism. In this section, we discuss the visual function and show an example of its use.

#### 2.5.4.1 Visual function: Inputs, Visual Mechanism, and Outputs

The inputs to the visual function come from the graph data. Also, they can be represented as predicate form. The output is a static visualization. The "between" step, that we call the Visual Mechanism, processes data to produce the visualization output.

The following are a couple examples of what could be a Visual Mechanism:

- Different layout options (force directed, circular, etc...) to display a graph *Michailidis* (2006)
- Coloring nodes by a rainbow gradient based on their degree value

Anything that results in a change on a display is fair game for the Visual Mechanism. What exactly happens at this stage is a combination of computation and visual encoding/rendering. The computational part is a set of composition functions and attribute creations on the graph. The visual rendering and encoding depends on software and hardware restrictions.

#### 2.5.4.2 Cytoscape Visual Display

A common visualization to create is coloring nodes on a rainbow gradient to show how an attribute value (in this case degree) varies on the graph. In this example, we show how this visual task occurs within a visual function in Cytoscape.

The input to the visual function is the phone data set from the VAST 2008 Challenge ( *Grinstein et al.* (2008), *Shaverdian et al.* (2009b)). The visual output we want to produce is a graph with its nodes colored based on their degree values.

The first step is to create the degree attribute on each node. We follow the composition function technique from Example II.2 to create the degree attribute.

In Cytoscape, this visual task corresponds to selecting a continuous color mapping on the degree attribute, Figure 2.13.A. Within the algebra, for each node we create a color attribute with a value that maps to its color gradient (computed through a mapping composition function). The algebra could create an attribute to characterize all the visual encoding and rendering features, but to keep this example brief we only mention the degree and color ones.

Next the visual encoding and rendering will produce the output. Since we cannot bound the software and hardware possibilities for different tools, within the algebra we note this as a $Tool\ Specific$ step. This step can be filled in with formalisms and languages that capture visual rendering specifics. Finally, the output layout is produced, Figure 2.13.B. In summary, this visual task corresponds to the following algebra:

1. Start: $\mathcal{V}_{input}\ \{PhoneDataSet1\}$

   A visual function is called on PhoneDataSet1.

2. $D_{phones} = \{G = (V, E),\ X = (X_V = X_{phone\_id}, X_{degree}, X_{color},\ X_E = (X_{date},\ X_{duration},\ X_{tower},\ X_{direction\_of\_call}),\ X_G = ())\}$

   After the user selects a mapping of color to degree, this color value is stored into $X_{color}$, a new node attribute.

3. $\mathcal{V}\ \{Tool\ Specific\}$

   A software process is run to produce the visualization output.

4. Complete: $\mathcal{V}_{output} \rightarrow Display$

   The visual function has completed with a display produced.

Setting a visual task into an algebra operator helps maintain the display information. In isolation, it seems like we took a simple visual task that might take an analyst a couple minutes to perform and turned it into a convoluted set of algebraic

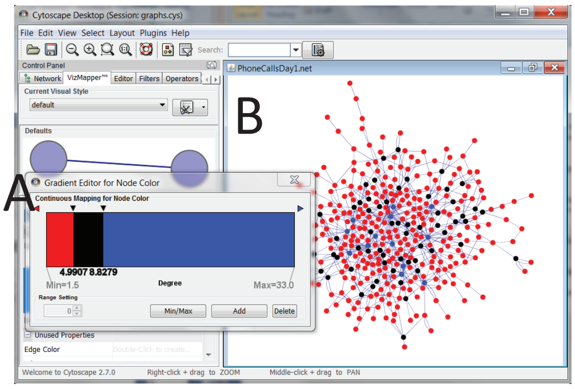

Cytoscape Desktop (Session: graphs.cys)
File Edit View Select Layout Plugins Help
Control Panel
Network VizMapper Editor Filters Operators
Current Visual Style
default
Defaults
A
B
PhoneCallsDay1.net
Gradient Editor for Node Color
Continuous Mapping for Node Color
4.9907 8.8279
Min=1.5
Degree
Max=33.0
Range Setting
Min/Max
Add
Delete
Unused Properties
Edge Color
Welcome to Cytoscape 2.7.0
Right-click + drag to ZOOM
Middle-click + drag to PAN

Figure 2.13: A continuous color gradient mapping to node degree is set on the phone data set from the VAST 2008 Challenge.

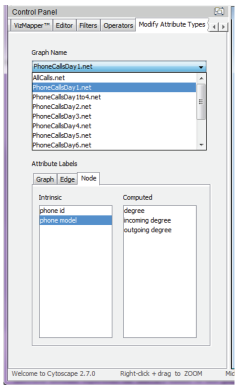
expressions. But actually it becomes even more convoluted if there is no algebra to track visual tasks. If several different visual tasks are completed to reach an analytical decision, it becomes difficult to replicate or verify the decision if the process is not documented. Also without using an algebra, as we discussed in the Introduction, comparing different tools and methods is difficult. Additionally, the visualization becomes more flexible. By breaking down the visualization task, we can change points of the task, and mix and match with different methods. For example, we can rerun this current example on a new graph by simply modifying the input.

## 2.6 Implementation Example in Cytoscape

In this section we utilize Cytoscape and various plugins to show how the theoretical concepts presented in the algebra are realized in a practical analytical tool.

### 2.6.1 Graph Model

Currently we have described the graph model that is composed of the graph, the attributes, and composition functions. Cytoscape already provides a lot of the implementation to display the model. Figure 2.15 shows a Cytoscape display window after some networks have been uploaded. The networks we have uploaded are from



AllCalls.net
PhoneCallsDay1.net
PhoneCallsDay1to4.net
PhoneCallsDay2.net
PhoneCallsDay3.net
PhoneCallsDay4.net
PhoneCallsDay5.net
PhoneCallsDay6.net

| Intrinsic | Computed |
|---|---|
| phone id | degree |
| phone model | incoming degree |
| | outgoing degree |

Figure 2.14: As part of our Visual Analytic Algebra, we define attributes as either computed or intrinsic. To allow the user to modify the type of an attribute we provide the following tab under the Control Panel. The user can view the attribute labels for nodes, edges, and the graph. To change the type from intrinsic to computed, the user can drag the attribute label from one type to another.

the VAST 2008 Challenge Data Set. The left panel shows the Control Panel. The Control Panel has several tabs.

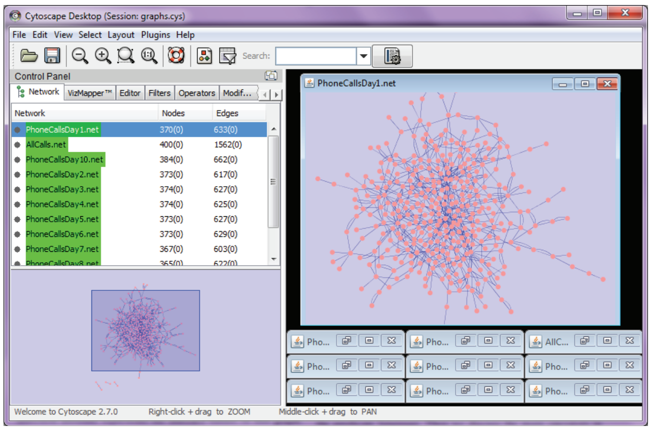
We have installed two plugins, Visual Analytic Algebra and NetMatch. These plugins have added extra tabs to this panel that we will introduce throughout the paper. The Network Tab displays a list of network names. The Modify Attribute Types Panel, shown in Figure 2.14, displays the attribute labels for graph, edges, and nodes. In this panel, the type of the attribute can be switched from intrinsic to computed type.

### 2.6.2 Predicate Language

Again we use Cytoscape to place a real analytical setting for some of the theoretical ideas presented in the previous subsections.

We illustrate a predicate with the NetMatch Query Editor. Using NetMatch, an analyst can draw a new predicate and place conditional values on its attributes. Figure 2.16 displays the NetMatch Query Editor.

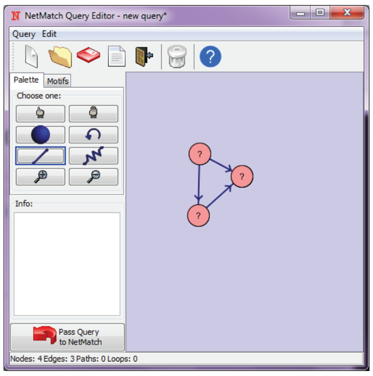
Next to produce a witness to the predicate, the analyst can click on the "Pass



| Network | Nodes | Edges |
|---|---|---|
| PhoneCallsDay1.net | 370(0) | 633(0) |
| AllCalls.net | 400(0) | 1562(0) |
| PhoneCallsDay10.net | 384(0) | 662(0) |
| PhoneCallsDay2.net | 373(0) | 617(0) |
| PhoneCallsDay3.net | 374(0) | 627(0) |
| PhoneCallsDay4.net | 374(0) | 625(0) |
| PhoneCallsDay5.net | 373(0) | 627(0) |
| PhoneCallsDay6.net | 373(0) | 629(0) |
| PhoneCallsDay7.net | 367(0) | 603(0) |
| PhoneCallsDay8.net | 365(0) | 622(0) |

Welcome to Cytoscape 2.7.0 Right-click + drag to ZOOM Middle-click + drag to PAN

Figure 2.15: The main Cytoscape window is shown. There are three main components to its design. The network panel displays the network. The data panel displays node and edge specific attribute information. The control panel has several tabs to perform different functions on the network. One of these tabs is the network tab, that shows all the networks opened during a current Cytoscape analysis session. It allows the user to switch between different networks by saving them under different names.



Nodes: 4 Edges: 3 Paths: 0 Loops: 0

Figure 2.16: The plugin NetMatch allows a user to draw a query. In essence, a query is a predicate. Attribute conditions are possible on nodes.

Query to NetMatch" button. This procedure will produce witnesses, as shown in Figure 2.17. Each witness will be displayed on its own row. The columns indicate the number of matches, the nodes that produced the match (corresponding to the mapping list), and an image of the witness.

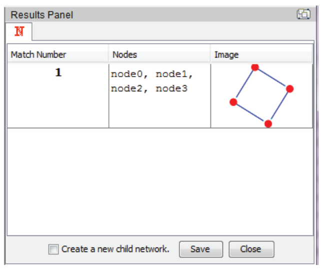


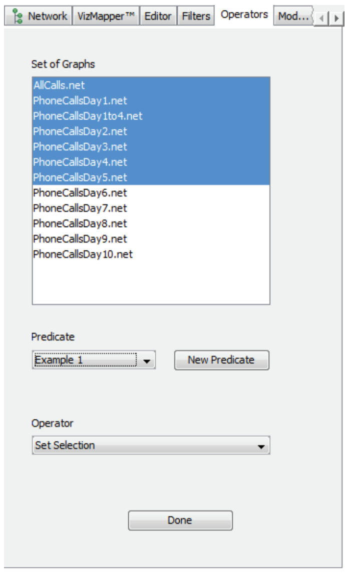
Figure 2.17: Using the NetMatch plugin, we show the result of matching the input graph and the predicate shown in Figure 2.5. The image shown is a witness found in the input graph. The Node column shows the mapping between the witness and the predicate. This figure is an example implementation of the Predicate, Witness, and Graph Matching features in the Visual Analytic Algebra.

### 2.6.3 Selection

We implement a direct way to perform set and element selection in Cytoscape through the Visual Analytic Graph Algebra Plugin. Under the Control Panel, the Operators tab has options to select a set of networks, the predicate, and operator to perform. Figure 2.18 this panel.

After a set selection has been applied on a set of the Phone Call Days data set from the VAST 2008 Challenge, the results are shown in Figure 2.19. The networks that satisfy the predicate are maximized and shown in the display panel. The networks that do not satisfy the figure are minimized in the background of the display panel.

To do an element selection, we use the same Operator tab. Figure 2.20 shows (1) the input graph, (2) the predicate, and (3) the results from element selection. As with a set selection, the analyst needs to first specify an input graph and predicate. Element selection then utilizes the graph matching function to produce results. Three witnesses are found that match the predicate. The analyst can then create new networks for the witnesses found by selecting the witness and checking "Create a new child network." The match information can be saved for future use.



Set of Graphs

AllCalls.net
PhoneCallsDay1.net
PhoneCallsDay1to4.net
PhoneCallsDay2.net
PhoneCallsDay3.net
PhoneCallsDay4.net
PhoneCallsDay5.net
PhoneCallsDay6.net
PhoneCallsDay7.net
PhoneCallsDay8.net
PhoneCallsDay9.net
PhoneCallsDay10.net

Predicate: Example 1

Operator: Set Selection

Figure 2.18: The Visual Analytic Graph Algebra Plugin also includes an Operators Tab in the Control Panel. Here the analyst can select a set of networks, the predicate list, and the operator to apply to the graphs. In this figure, the set selection Operator has been selected with the Predicate from Figure 2.1.

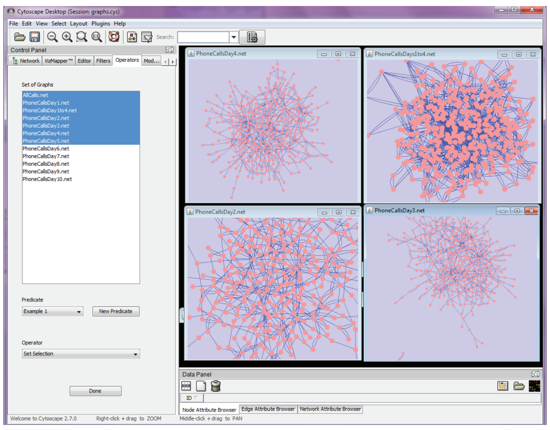
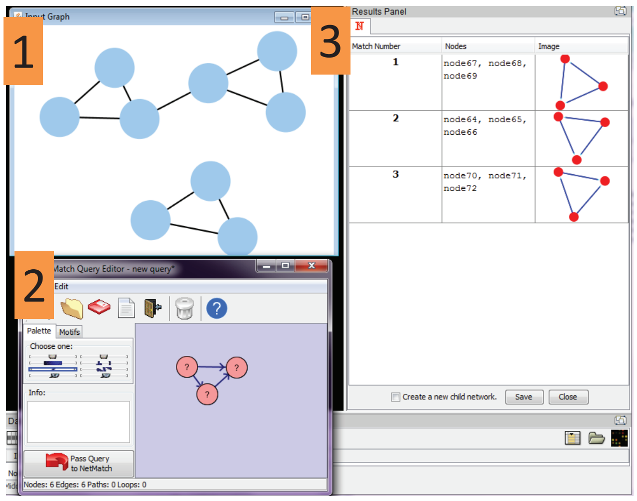

Cytoscape Desktop (Session: graphs.cys)
File Edit View Select Layout Plugins Help
Control Panel
PhoneCallsDay4.net
PhoneCallsDays1to4.net
PhoneCallsDay2.net
PhoneCallsDay3.net
Data Panel
Node Attribute Browser
Edge Attribute Browser
Network Attribute Browser
Welcome to Cytoscape 2.7.0
Right-click + drag to ZOOM
Middle-click + drag to PAN

Figure 2.19: This figure shows an example implementation of set selection in Cytoscape. Once the "Done" button has been clicked, the result of the set selection is the set of graphs that satisfy the predicate are opened and displayed in the Display Panel. In this figure, four graphs are displayed.


1
Input Graph
2
Match Query Editor - new query*
Edit
Palette
Motifs
Choose one:
Info:
Pass Query to NetMatch
Nodes: 6 Edges: 6 Paths: 0 Loops: 0
3
Results Panel
Create a new child network.
Save
Close

| Match Number | Nodes | Image |
|---|---|---|
| 1 | node67, node68, node69 | |
| 2 | node64, node65, node66 | |
| 3 | node70, node71, node72 | |

Figure 2.20: This figure shows (1) the input graph, (2) the predicate, and (3) the results from element selection.

### 2.6.4 Aggregation

In Cytoscape to do an aggregation, an analyst starts at the Operators tab. Figure 2.21 shows a complete list of the operator options available.

Under the hood, a set aggregation has the following components: a set selection, followed by a union and update on the computed attributes. An element aggregation creates new merged nodes within the graph. Figure 2.22 and Figure 2.23 are element aggregations by all and per structures done in Cytoscape, respectively. We use the same input and predicate as in Figure 2.10.

Since Figure 2.22 is an aggregation by all structures and all nodes in the input graph are witnesses to the predicate, they become merged into a single supernode. In Cytoscape the model witness is displayed within the supernode. This feature helps distinguish supernodes from lower granularity nodes. Figure 2.23 has the same aggregation by structure result as shown in Figure 2.10. In Cytoscape, the aggregated graphs are created as ”child” graphs to the input graph. In the network tab, the child graphs will be listed below their parent graph.

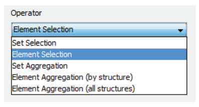

Operator
Element Selection
Set Selection
Element Selection
Set Aggregation
Element Aggregation (by structure)
Element Aggregation (all structures)

Figure 2.21: On the Operators Tab, the drop down menu shown has all of the operators available for easy access for the analyst.

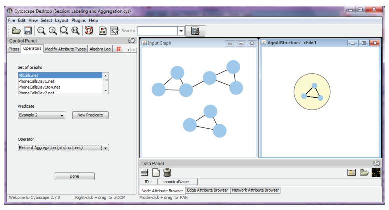


Figure 2.22: A Cytoscape Aggregation by All Structures is performed on the input graph shown on the left.

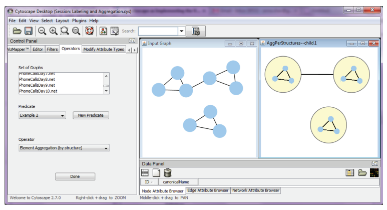


Figure 2.23: A Cytoscape Aggregation Per Structures is performed on the input graph shown on the left.

## 2.7 Experiment

A central motivation for the graph algebra is to support reproducibility of analysis. In this section, we evaluate this experimentally with a user study. First we define reproducibility and how it can be measured given an experimental analysis.

### 2.7.1 The Reproducibility Metric

We measure reproducibility by comparing the similarity between the *target* and the *copy* graph. The target graph is created when analyst $A$ performs manipulations on a *start graph*. The recorded manipulations of analyst $A$ is the *analysis* used to create the target graph.

When analyst $B$ is given $A$'s analysis and the *start graph*, the graph she produces by following the instructions is the *copy graph*. Since she cannot see the *target* graph, her copy may have errors depending on how well she followed the instructions and how well the analysis is written. Therefore, reproducibility reflects the ease for other analysts to repeat an analysis on the same or different data sets.

We calculate reproducibility with the following metric:

$$Reproducibility = \begin{cases} \frac{T-M}{T} & \text{Copy does not contain extra elements} \\ 0 & \text{Otherwise} \end{cases}$$

where,

$T$ = number of elements in the target graph

$M$ = number of missing elements in copy graph

We calculate the total number of elements in the target graph by summing the cardinality of the node and edge sets. The reproducibility metric is computed per graph. A higher value is a better reproducibility. Figure 2.24 shows a *target* and *copy* graph. The copy has 3 extra graph elements, namely, the circle and edges between the square and the lower left circle. The reproducibility is equal to 0.

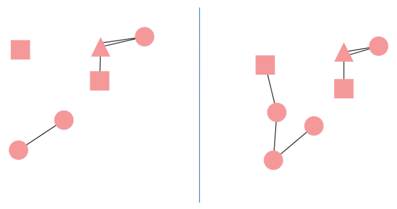


Figure 2.24: The graph on the left is what the final graph after the analysis should be. The graph on the right shows one of the graphs produced during the user study.

### 2.7.2 Procedure

We test the reproducibility of an analysis reported in plain English versus in the algebra. Since there is not a universal systematic language that defines the basic graph data manipulation actions in visual analytics, English is usually used to share results. So, we compare sharing an analysis in English versus the Visual Analytic Graph Algebra.

The user study has two phases. In the first phase we show the start and summarized graph to 10 different users. All users are graduate students at the University of Michigan. Figure 2.25 shows the two graphs. The tasks involve only element-level operators. Half of these users' task is to write in English the set of transformations to the start graph to create the target graph. This set of transformations is an English *analysis*. The other users are given algebra operator training. Their task is to create an algebraic *analysis* for the target graph.

After phase one, we have 5 analysis written in English and 5 analysis written in the visual analytic algebra. In the second phase, we give one of these analyses and the start graph to 10 different users. Their task is to read the *analysis* and perform its transformations on the start graph. The users who are given an algebra analysis are trained on the algebra operators. We compute the reproducibility metric on the replicated graphs.

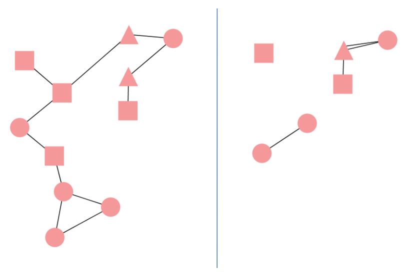


Figure 2.25: Starting from the graph on the left, the graph on the right is created, the summarized graph.

### 2.7.3 Results and Analysis

The study was performed using pen and paper. We chose a graph with less than 10 nodes to make drawing and transforming the graph realistic for users. After experimenting with several different graphs, we selected the graph pair shown in Figure 2.25. We wanted users to quickly spot differences between the graphs without writing extremely naive analysis.

The graph algebra will not likely be used directly by analysts. It is a universal language between tools and methods that can be mapped to high level tool operations. For example, in Cytoscape, creating a predicate is a drag-and-click process versus writing a complete algebraic predicate tuple. Since the algebra implementation in Cytoscape is within its early prototype stage, we want to separate noise from the users' system experiences from testing whether they are empowered or overwhelmed by the algebra. So we bypass system tools in the user study to test the algebra with pen and paper. In practice, the tool designer would need to be aware of the algebra and how the tool operations map to the algebra operations.

The following is an example of an English analysis written: "Start from the upper left corner, delete the square and move diagonally down and delete the circle and square and the circle again. Then take out the upper colored-in triangle and an edge from the triangle to the circle." The English analyses utilized node placement and deletions in the directions. The algebra analyses were more specific: creating

| English | Algebra |
|---|---|
| 0.7 | 1 |
| 0.7 | 1 |
| 0.5 | 0.6 |
| 0.5 | 0.6 |
| 0 | 0 |

Table 2.1: This table shows the copy graphs' reproducibility sorted from best to worst.

attributes, selections, aggregations. Users were trained on the algebra for 15 minutes. They were all familiar with graphs and had previously taken a programming course.

We computed the *reproducibility* for the copy graphs, shown in Table 2.1. The algebra has higher reproducibility in the copy graphs. Based on the p-value = 0.02573 from two sample t-test, the algebra group has statistically significant higher average score than the English group. These results are reasonable because the algebra provides a precise language for the analysts.

## 2.8 Conclusions

In this study, we introduce a framework and the corresponding algebra for visual analysis of graph based data sets. We provide a detailed description of its operators and illustrated its use through an implementation in Cytoscape.

As already pointed out, the proposed algebra provides a rigorous workflow model for understanding and documenting the visual analytics process that the users undertake when exploring their data. However, as graphs become larger and the list of their node and edge attribute grows, the need for fast responses to users' selections and manipulations of the data becomes more prevalent. The rigorous structure of our visual algebra allows for optimizing the performance of its constituent operators, while incorporating constraints imposed by visualization (e.g. screen size), thus accelerating the visual exploration and discovery process. This constitutes a topic of current research.

Although we provide an expressive graph manipulation language through predicates and operators, there are limitations to the language. For example, noisy and incomplete data is a piece to any analytical problem. Expanding the algebra to effectively handle these types of data is a promising future direction.

# CHAPTER III

# Querying Graphs with Uncertain Predicates

## 3.1 Introduction to Uncertainty model

In many real world applications, a scientist may have some idea about what values and structure she expects. But to manually track her uncertainty from the conception of the uncertain sub-network query to the multiple results is a demanding task. This management problem is even more confounding when there are multiple stages in network analysis. At any point in the workflow, there may have been an uncertain query. The analyst has to manually determine how likely a resulting graph represents the queries she made over the whole workflow. One motivating application occurs in social network intelligence. In the 2009 VAST Challenge (*VAST* (2009)), two hypothetical social networks are described with uncertainty on degree, connectivity, and attribute values. The challenge was to determine the more likely social network based on the graphs returned by the different hypothetical social networks.

In this chapter, we provide a model for the uncertain predicate in a network query setting. This model guides the user to define the uncertain query. Then it generates results, with ranking based on the user's preferences. We build the uncertainty querying over an existing visual analytic algebra. We show how the uncertainty on the queries can be composed over an analysis workflow. Our uncertainty model follows this basic outline: First a user inputs an uncertain query. Next the system expands this query into a set of exact queries representing the uncertain query. Then the user places a probability on each exact query which notes how well the exact query captures the uncertain query. The system then computes the graph manipulation operation and probabilities for the results.

In the uncertain query problem we assume the graph itself is exact. The uncertainty remains only at the query level. The rank value associated with the result of an operator is saved as a computed graph attribute and denotes how well the outcome describes the user's uncertain query. However, this value does not introduce uncertainty on the graph elements and attributes' existence.

## 3.2 Uncertain Predicate Language

Based on our algebra framework, an exact predicate tuple is given by $\rho = (V, E, \mathbb{X}_V, \mathbb{X}_E, \mathbb{X}_G, !E)$. We expand the predicate language to specify attributes and edges which contain uncertainty.

For attribute uncertainty, the condition lists are modified to include an uncertainty language. The condition lists, $\mathbb{X}_V$, $\mathbb{X}_E$ , and $\mathbb{X}_G$, in the predicate can have conditions now of the form "attribute_name $\approx$ value", where $\approx$ denotes that there exists uncertainty on this attribute. Similarly, edge existence uncertainty is denoted with an $\approx$ prefix over an edge in the edge list, $E$. Now that we provide the user a way to describe the uncertain attributes and connectivity, we can convert the uncertain predicate to a set of exact predicates.

## 3.3 Process of Turning an Uncertain Query to an Exact One

Given an uncertain predicate, $\alpha^*$, either the user or the system generates a set of non-overlapping exact predicates, $\{\alpha_1, \ldots, \alpha_j\}$ that characterize the uncertain predicate. For any $\alpha_i$ and $\alpha_j$ where $i \neq j$ in the exact predicate set, the non-overlapping condition strongly enforces that $\alpha_i \neq \alpha_j$. If the system performs this mapping, pre-defined substructure similarity measures can be used. In the case of numerical attributes, Euclidean distances can be used, with a simple threshold dissimilarity.

Next the system requests the user to assign a probability to each element in this set of exact predicates. This probability corresponds to how well the exact predicate captures the uncertain predicate structure. We denote the probability set $\Omega_{\alpha^*} = \{p_{\alpha_1}, \cdots, p_{\alpha_j}\}$ where $p_{\alpha_i}$ is the probability for exact predicate $\alpha_i$ with respect to uncertain predicate $\alpha^*$ based on the graph at the time. Each $p_i$ follows a Bernoulli distribution with the probability of success equal to $P(\alpha_i = \alpha^*)$; therefore, each $p_{\alpha_i}$ follows a complete disjoint distribution. $\sum_i p_{\alpha_i}$ may not equal to one.

This procedure is illustrated below with the following examples.

**Example III.1.** Given an uncertain predicate $\alpha^* = \{V = v_1, X_{v_1.age} \approx 4\}$ and a friendship network $G_{D_0} \in D_0$, a user is interested in selecting people (i.e. nodes) in the network with age around 4. We can create a computed attribute for the range of the age attributes. In this example the range of the age attributes is from 2 to 6. The system returns a set of exact predicates within a distance. For example, if we set the threshold distance as a quarter of the variable range, 1, the system will return the following exact predicates:

$$\alpha_1 = \{V = v_1, X_{v_1.age} = 3\}$$
$$\alpha_2 = \{V = v_1, X_{v_1.age} = 4\}$$
$$\alpha_3 = \{V = v_1, X_{v_1.age} = 5\}$$

Based on this set, the user will define the probability set $\Omega_{\alpha*} = \{p_{\alpha_1}, p_{\alpha_2}, p_{\alpha_3}\}$ where $p_{\alpha_i} = P(\alpha_i \text{ correctly describes } \alpha^*)$ and all $p_{\alpha_i}$s are independent from others.

**Example III.2.** Figure 3.1 shows a predicate with uncertainty on the node's color attribute and a graph $G_0$. A set of exact predicates can be generated by using the

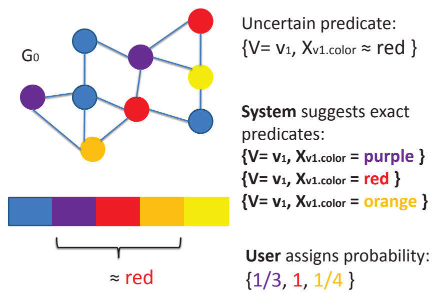

G0
Uncertain predicate:
{V= v1, Xv1.color ≈ red }
System suggests exact predicates:
{V= v1, Xv1.color = purple }
{V= v1, Xv1.color = red }
{V= v1, Xv1.color = orange }
User assigns probability:
{1/3, 1, 1/4 }
≈ red

Figure 3.1: Example to transform a categorical uncertain predicate into a set of exact predicates.

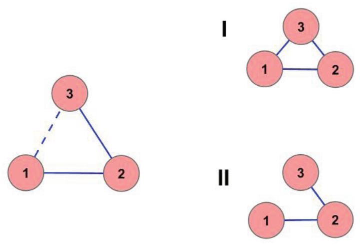


Figure 3.2: Example of edge certainty. Dash line stands for uncertain connections

$G_0$'s overall color range.

**Example III.3.** Given an uncertain predicate $\beta^* = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, \approx e_{13}\}, x_{v1.age} = 3\}$, there are two suggested exact predicates in the set: $\beta_1$ as $\{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, x_{v1.age} = 3\}$ (which suggests a fully connected graph), and $\beta_2$ as $\{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}\}, x_{v1.age} = 3, !e_{13}\}$ (which suggests disconnected network) shown in Figure 3.2.

Instead of attaching confidence scores to the output graphs once they are returned, we allow users to define the probabilities at an earlier stage. By defining probabilities at the predicate level before the matching function is called, the user is given more flexibility and control over the results. For example, the user can assign zero probability to unrelated exact predicates. This setup significantly improves the efficiency of the system by limiting unnecessary matchings. If the user does not assign any probability values to the exact predicate set, the system will assume the same probability value, one, to all exact predicate structures. It implies that all exact predicates in the set can correctly capture the given uncertain predicate.

Furthermore, by attaching probabilities to only a few exact predicates rather than the entire possible output set for a given uncertain query, we can reduce the workload for users in assigning probability values. Moreover, in order to further reduce the computation complexity, the system can force either the size of the exact predicate set or the number of output graphs to be below a predefined threshold.

## 3.4 Solution Generation

After we transform the uncertain predicate into a set of exact predicates with user defined probabilities, we can solve the uncertain querying problem by performing the operator on each exact non-overlapping predicate structure separately.

For each exact predicate, $\alpha_i$, we call the operator given the input graph $D_i$. For each of the output graphs, $D'_j$, we create an operator distribution set, $\Omega_{D'_j|D_i}$. By construction, the length of the operator distribution list will equal to number of exact predicates. If the system can discover at least one witness structure based on the exact predicate $\alpha_s$ in the output graph $D'_j$, the $s^{th}$ term of the operator distribution set will equal the user defined probability, $p_{\alpha_s}$. On the other hand, if the system returns an empty list after applying the operator based on predicate $\alpha_t$, the $t^{th}$ term of the operator distribution set will equal zero. The zero/ fraction patterns in the operator distribution presents a summarization of the outcome of each exact predicate given the input graph.

Output graph structures depend on the operator along with its predicate conditions and input graphs. Output graph probabilities need to reflect probability information from all three components as well. The internal operator distribution set carries probability information from both the operator and its predicate conditions, while the probability distribution set, for each input graph, contains necessary probability information on the input graph set. In order to compute the output graph probability, one needs to determine the output probability distribution set, the first sublist in the probability list. The construction of the output probability distribution set is based on both the operator distribution set, $\Omega_{D'_j|D_i}$, and the input probability distribution set, $\Omega_{D_i}$.

**Definition III.4.** The output graph probability distribution set is defined as the tensor product of the user defined operator distribution set and the input probability distribution set.

In mathematics, the tensor product is denoted as $\otimes$. It may be applied to vectors, matrices. In this study, we are only interested in the tensor product of two vectors. For example, the tensor product of vectors $\vec{a}$=$(a_1, a_2)$ and $\vec{b}$=$(b_1, b_2, b_3)$ is a 6-dimensional vector, $(a_1b_1$,$a_1b_2$, $a_1b_3$,$a_2b_1$, $a_2b_2$,$a_2b_3)$. Based on the independence assumption among exact predicate distributions, the output probability distribution lives in the product space of the input probability distribution set and the operator distribution set. The tensor product provides us the technique to keep track of the detailed probability information for each output graph. Based on the output probability distribution set, $\Omega_{D'_j|D_i}$, we can find output probabilities, $P_{D'_j}$, for each output graph.

**Definition III.5.** The output graph probability for an analytic step based on uncertain predicate $\alpha^*$ is defined as follows:

$$\begin{aligned} P_{D'_j} &= 1 - P(D'_j \text{ contain no witnesses for } \alpha*) \\ &= 1 - P(\text{no witness for} \alpha_1) \cdots P(\text{no witness for } \alpha_k) \end{aligned}$$

where $P(\text{no witness for } \alpha_i)$ is one minus the $i^{th}$ term found in the output probability distribution set,$\Omega_{D'_j}$, based on definition.

Since all probabilities follow different independent distributions in the output probability distribution set, we can break down the joint probability into a product of multiple user defined probabilities. The probability of this particular output can be interpreted as the probability that an output contains at least one matching for predicates $\alpha_1$ to $\alpha_k$. The detailed calculation will be further demonstrated in the examples below.

**Example III.6.** This example describes Figure 5.1 which illustrates an element selection operator, given an uncertain predicate on input graph $D$ with a probability list as $\{[a, b], [1-(1-a)(1-b)]\}$. The uncertain predicate is: $\alpha^*$=$\{V = \{v_1, v_2, v_3\}$, $E = \{e_{12}, e_{23},\ e_{13}\}$, $X_{v_1.\text{age}} \approx 4\}$. An element selection given $\alpha^*$ will return a triangle structure of nodes where at least one node has age attribute $\approx 4$. We detail the entire element selection process given this uncertain predicate.

1. The system generates a set of exact predicates. The exact predicates are shown in Figure 5.1 part B:

$$\begin{aligned}
\alpha_1 &= \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v_1.\text{age}} = 3\} \\
\alpha_2 &= \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v_1.\text{age}} = 4\} \\
\alpha_3 &= \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v_1.\text{age}} = 5\}
\end{aligned}$$

2. The user assigns probabilities for each of the exact predicates in part B. $\Omega_{\alpha*} = \{\frac{1}{3}, 1, \frac{1}{2}\}$, where $\alpha^*$ can be described by $\alpha_1$ and $\alpha_3$ correctly $\frac{1}{3}$ and $\frac{1}{2}$ of the time, while $\alpha^*$ can be captured by the condition listed in $\alpha_2$ completely.

3. Next, the element selection operator based on exact predicate is called three times for each exact predicate. For each operator evaluation, the element selection operator returns output graphs. Since each of the output lists only satisfies one exact predicate, the probability of the output list equals to the probability of the exact predicate in this example. This step is shown in Figure 5.1 part C.

4. Finally, the operator distribution set is calculated for each output graph. We perform a set union on the three output lists. In effect, output graphs with

identical graph structure and node id's are merged. Next we compute the operator distribution set for each unique output graph. The operator distribution set is shown in Figure 3.3 part D.

5. The output probability distribution set is updated for each output graph by using the tensor product on the input probability distribution set $[a, b]$ and the operator distribution sets. The output probabilities are calculated as follows:

$$P(D'_1) = 1 - (1 - \frac{a}{3})$$
$$P(D'_2) = 1 - (1 - \frac{a}{3})(1 - \frac{a}{2})(1 - \frac{b}{3})(1 - \frac{b}{2})$$
$$P(D'_3) = 1 -$$
$$(1 - \frac{a}{3})(1 - \frac{a}{2})(1 - a)(1 - \frac{b}{3})(1 - \frac{b}{2})(1 - b)$$

Since all the output probabilities are constructed based on the probability distribution set which corresponds to the correctness of capturing the uncertain query and its input's probability distribution set, we are able to compare all the outputs from the same operator. In this example, the graph , $D'_3$, with the highest output probability does the best job in capturing the users' target structure in the element selection step. Moreover, we treat the newly computed probability list for each output as a computed graph attribute. The visual algebra stays closed, since both inputs and outputs are defined to be the same type of objects, an attributed graph set.

## 3.5 Uncertainty Model for Complex Structural Predicates

In the previous sections, we introduced the uncertainty model to handle simple uncertain predicates. In this section, we apply an operator on a complex uncertain predicate. The complex structural uncertain predicate is an uncertain predicate which contains more than one uncertainty, and it appears as a joint predicate with multiple

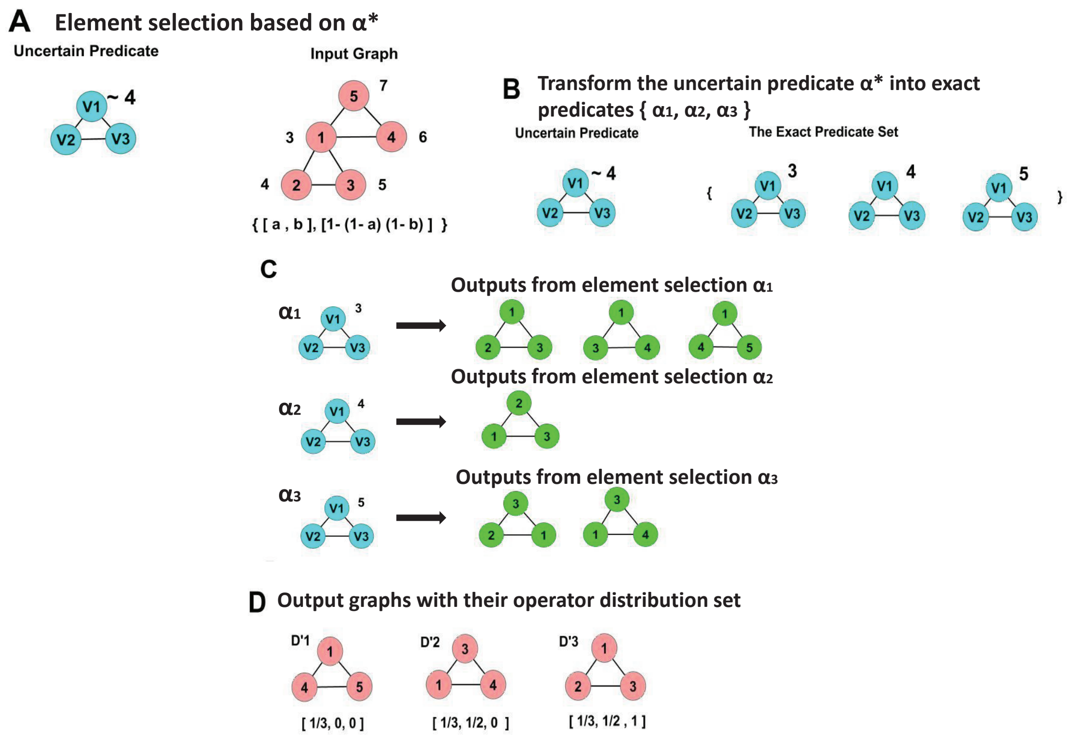

A Element selection based on α*
Uncertain Predicate
Input Graph
{ [ a , b ], [1- (1- a) (1- b) ] }
B Transform the uncertain predicate α* into exact predicates { α1, α2, α3 }
Uncertain Predicate
The Exact Predicate Set
C
Outputs from element selection α1
Outputs from element selection α2
Outputs from element selection α3
D Output graphs with their operator distribution set
D'1
D'2
D'3
[ 1/3, 0, 0 ]
[ 1/3, 1/2, 0 ]
[ 1/3, 1/2 , 1 ]

Figure 3.3: Example of element selection based on an uncertain predicate $\alpha*$ in blue. The numbers inside the node denote the id. The numbers outside the node denote the age attribute's value. The different colors are used for visualization clarity. The pink represents input and output graphs. The blue represents predicates. And the light green represents witnesses.

predicates combined with logical functions, such as AND and OR. Next, we would like to simplify the complex structural predicate problem into a few operators based only on simple predicates. This simplification will significantly reduce the user's workload for defining the probability distribution of the exact predicate set.

There exists no general solution to simplify the aggregation operator based on a complex structural predicate into a few smaller fractional structures and still persevere the same output, because aggregation operators, both element and set, may change the structural information of the qualified outputs even before the usage of uncertain querying model.

**Example III.7.** This example describes Figure 3.4 which illustrates an element aggregation operator given a complex structure predicate on graph $D$. The complex predicate list contains two exact predicates: $\alpha$=$\{\alpha_1$ AND $\alpha_2\}$ where $\alpha_1 = \{v_1$, $X_{v1.color} = Red\}$ is a red colored node, $\alpha_2 = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}\}$ is three connected nodes. An element aggregation operator based on the complex predicate, $\alpha$, for the given $D$ will return only one node as in Figure 3.4 output I, since we merge all red nodes and triangular structures all together into the same group ,node $v_8$ , defined by $\alpha_1$ and $\alpha_2$. However, if we simply transform the complex step into two smaller steps, where we apply $\alpha_2$ first and $\alpha_1$ second. The system will return different outputs as in Figure 3.4 output II, since we merge all red nodes in $D$ to create a super node, $v_9$, in the first step will destroy the triangular structure on the left. As in figure 3.4, all newly created super nodes are pointed by nodes

On the other hand, selection operators based on a complex predicate list can be broken into smaller fractions, while preserving the same outputs; therefore, we may be able to simplify the complex structural uncertain selection problem under some predefined assumptions. In the uncertainty model for a simple predicate, probabilities of the final outputs from complex uncertain ones are still constructed from the output probability distribution set. Since we are still only dealing with a single operator

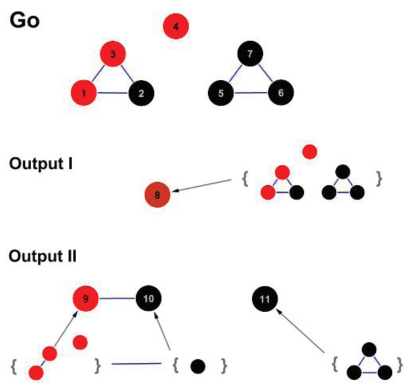


Figure 3.4: Example of element aggregation.

at this time, the operator distribution is still constructed based on the probability distribution of the exact predicate set. By definition, a complex structural uncertain predicate contains multiple uncertain variables, attributes or structures; hence, the probability distribution of the exact predicate set will live in a multidimensional space, one for each uncertain variable. In other words, the probability distribution of the exact predicate set is a high dimensional joint probability for multiple uncertain variables. Therefore, by providing additional assumptions for the high dimensional probability distribution, we are able to break the high dimensional joint distribution into smaller marginal or conditional independent probability distributions. Users will be able to work only in the low dimensional space and consider the probability to describe the behavior of one uncertain variable at a time.

If there exists some degree of dependence among uncertain variables within the complex uncertain predicate, the high dimensional probability of the complex predicate could be further broken into conditional independent components with some constraints. For example, a user tries to select people with approximately high income (I), college education level (E) and with high percentage of house ownership (O), where three uncertain variables tend to be highly correlated, the user cannot simply break the probability of exact predicates which describes the uncertain predicate into to independent components. However, instead of using only the joint probability with three variables, $P(I \text{ AND } E \text{ AND } O)$, the system can reduce the user's workload by allowing the conditional probabilities, $P(E)$, $P(I|E)$ and $P(O|I)$. In this case, the house ownership is independent from education level given income information, but in order to keep the consistence over selection operators the defined conditional probabilities needs to satisfy $P(I \text{ AND } E \text{ AND } O) = P(E)P(I|E)P(O|I)$ constraint. The reduction of the user's workload can be easily explained in the follow situation. For example, if for each of the uncertain variable, $\approx$ high income, $\approx$ college education level and $\approx$high percentage of house ownership, can be transformed to three exact values, such as $\approx$ college education level equals to Master degree(MA), Bachelor degree(BA) and Associate degree(AA), the user needs to define probabilities for 27 elements in the high dimensional case, but only 9 probability values in the conditional independent case under the correct constraint.

If a complex uncertain predicate can be broken into smaller independent structures, the probability space of the complicated predicate is equal to the product space of the probabilities of the smaller uncertain components,$\Omega_{\alpha^* AND \beta^*}$=$\Omega_{\alpha *} \otimes \Omega_{\beta *}$. This fact is supported by the independence between each uncertain predicate designed by the user. By the commutative property of multiplication, we can reorder the composed selections and preserve the same output probabilities. Figure 3.5 illustrates how a composed element selection can be broken into two components. Since independent selection step based on simple uncertain predicate can be reorder and still persevere the same output with consistent output probability, we can pick the optimal selection ordering to minimize the system computational complexity. This special property of the selection operator presents the opportunity to use rewrite rules for optimization.

**Example III.8.** To illustrate the usage of the uncertainty querying method in the independent case, we present the following example. The study has a complex uncertain predicate with a triangle node structure with uncertainty on two attributes, shown in Figure 3.7. We show the element selection process twice. The first time, we

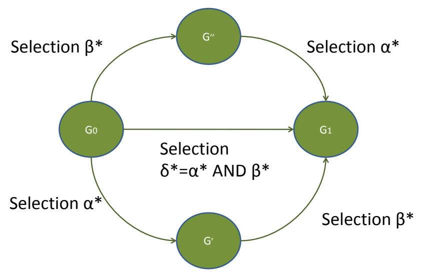


Figure 3.5: Example of joint element selection. The selection operator with uncertain predicate $\alpha*$ AND $\beta*$ is split into two. One split selection has uncertain predicate $\alpha*$ and the other has uncertain predicate $\beta*$

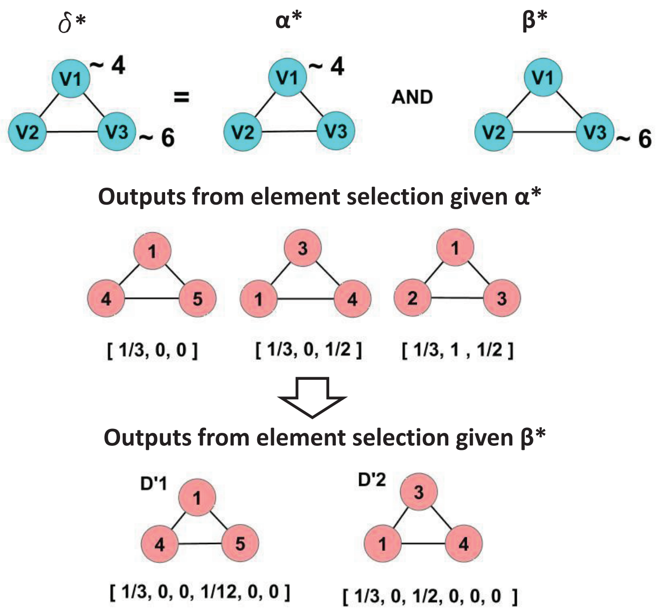


Figure 3.6: This figure shows that we get the same outputs as the combined uncertain predicate structure if we split the predicate structure into $\alpha*$ and $\beta*$ and perform two serial element selections.

call element selection with the predicate $\delta^* = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v1.age} \approx 4, X_{v3.age} \approx 6\}$ shown in Figure 3.7. The second time, we call two element selections in a serial fashion. The first with predicate $\alpha^* = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v1.age} \approx 4\}$; and the second with predicate $\beta^* = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v3.age} \approx 6\}$(shown in Figure 3.6). The latter, two element selection calls, will produce the same outputs as the single element selection call with predicate $\delta^*$.

Figure 3.7 part A shows an attributed graph $D$ with the input probability list as $\{\{1\},\{1\}\}$. This graph represents a social network of friends where nodes are people and edges represent a friendship between two people. The numbers inside the nodes represent an id; the numbers outside the node are the age attribute's value. Graph $D$ has the identical graph structure we use in the examples throughout the paper.

A user would like to call an element selection $\sigma_{element,\delta^*}(D)$, with the uncertain predicate $\delta^*$, shown in Figure 3.7 part A. The system generates a set of exact predicates in Figure 3.7 part B,

$$\delta_1 = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v1.age} = 3, X_{v3.age} = 6\}$$
$$\delta_2 = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v1.age} = 4, X_{v3.age} = 6\}$$
$$\delta_3 = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v1.age} = 5, X_{v3.age} = 6\}$$
$$\delta_4 = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v1.age} = 3, X_{v3.age} = 7\}$$
$$\delta_5 = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v1.age} = 4, X_{v3.age} = 7\}$$
$$\delta_6 = \{\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{13}\}, X_{v1.age} = 5, X_{v3.age} = 7\}$$

and requests user to input 6 probabilities for exact predicates, $\Omega_{\delta^*} = \{\frac{1}{3},\ 1,\ \frac{1}{2}, \frac{1}{12},$ $\frac{1}{4},\ \frac{1}{8}\}$, corresponding to the uncertain predicate. Using the process outlined in the pervious section, element selection based on the exact predicates generates outputs for each of the exact predicates shown in Figure 3.7 part C. For the input graph $D$, we can only find witness for element selection based on $\delta_1$, $\delta_3$ and $\delta_4$. All other exact matching procedures return an empty witness list after applying the graph matching function $\gamma$. Finally, the system outputs the graphs and together with their probability lists shown in Figure 3.7 part D. There are two final outputs with output probabilities, $P(D'_1) = \frac{7}{18}$ and $P(D'_2) = \frac{2}{3}$.

With this example, we now show if we decompose the complex uncertain predicate to the following simple predicates: $\alpha^*$ equals to $\{V = \{v_1,v_2,\ v_3\},\ E=\{e_{12},$

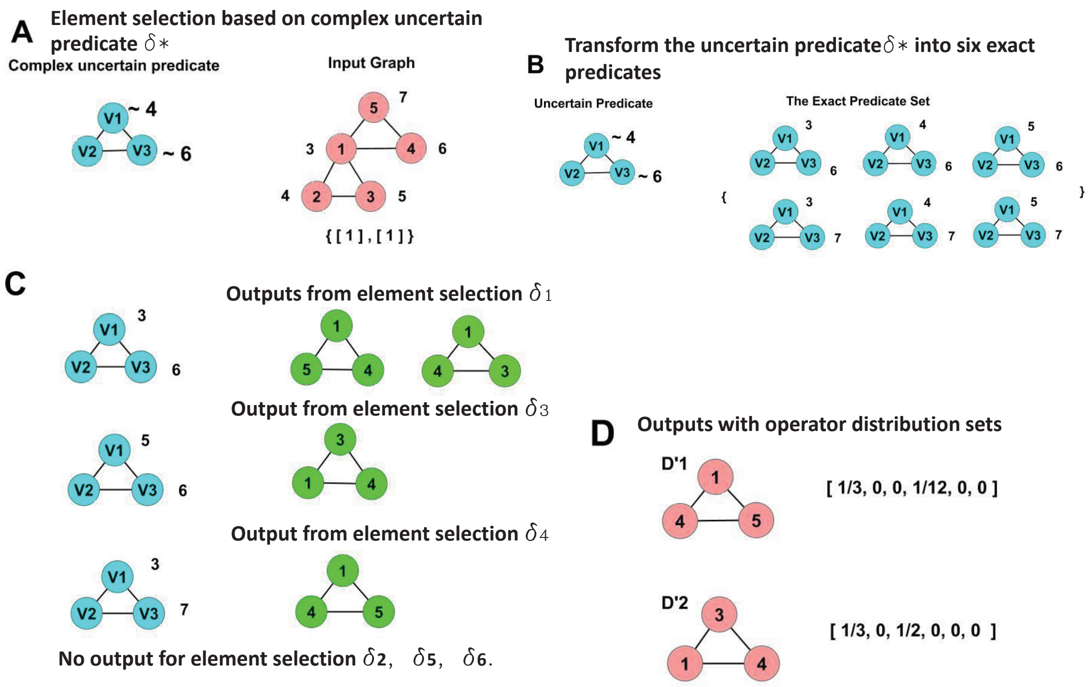

A Element selection based on complex uncertain predicate δ∗
Complex uncertain predicate
Input Graph
{[1],[1]}
B Transform the uncertain predicate δ∗ into six exact predicates
Uncertain Predicate
The Exact Predicate Set
C
Outputs from element selection δ1
Output from element selection δ3
Output from element selection δ4
No output for element selection δ2, δ5, δ6.
D Outputs with operator distribution sets
D'1
[ 1/3, 0, 0, 1/12, 0, 0 ]
D'2
[ 1/3, 0, 1/2, 0, 0, 0 ]

Figure 3.7: This example shows a complicated uncertain predicate structure with two attribute value uncertainties. Part A shows the uncertain predicate and the attributed graph $D$. Part B shows the uncertain predicate transform into a set of six exact predicates. Part C shows the output from exact element selection. Finally, Part D shows the final output with its operator distribution set. The numbers inside the node denote the id. The numbers outside the node denote the age attribute's value. The different colors are used for visualization clarity. The pink represents input and outputs graphs. The blue represents predicates. And the light green represents outputs from exact matching problems.

$e_{23}$,$e_{13}\}$,$X_{v_1.\text{age}} \approx 4\}$ as in the pervious example and $\beta^*$ equals to $\{V = \{v_1,v_2, v_3\}$, $E = \{e_{12}, e_{23},\ e_{13}\}$,$X_{v_1.\text{age}} \approx 6\}$, and perform serial element selections, we receive the same outputs with the same output probabilities as before. Figure 3.7 shows an element selection with uncertain predicate $\alpha^*$ called first and its outputs. Next, an element selection with uncertain predicate $\beta^*$ is called. The outputs in Figure 3.7 part C and the outputs generated by $\beta^*$ (after $\alpha^*$ is called) are the same. By applying serial element selection over one complicated selection step, we effectively cut down the user's workload from defining a probability set $\Omega_{\delta *}$ which contains 6 probability scores to construct two simple probability sets $\Omega_{\alpha *}$=$\{\frac{1}{3}, 1, \frac{1}{2}\}$ and $\Omega_{\beta *}$=$\{1, \frac{1}{4}\}$, where $\Omega_{\delta *} = \Omega_{\alpha *} \otimes \Omega_{\beta *}$ constrain ensure the consistences in output probabilities.

## 3.6 Visual Analytic Work Process

In real analysis applications, multiple operators are used to manipulate the input graphs and produce final solutions. In other words, the final outputs often resulted from composition of multiple operators, one for each analytic step. Since we have defined an uncertainty querying model for a single operator step based on the existing visual analytic algebra framework, we are going to expand the single step model to handle the multi-step process.

### 3.6.1 Model for Composition of Operators

Before dealing with the construction of the uncertainty model to capture how probabilities of multiple uncertain predicates are composed during an analysis directly, we first define the composition of operators. The composition of operators is a composition process which often contains multiple operators to capture the users' preferences during an analytic process. Since our uncertainty model is constructed based on our existing visual analytic algebra framework, where we adopt existing visualization systems to help and guide data analysis, calling and utilizing visualization functions during the analysis is a key step during a visual analytic process. By the default setting in most existing visualization system is that users will automatically receive a new visual representation of the data after each manipulation; therefore, we assume that all operator steps in the process are separated by the usage of the visualization function, so all consecutive operators cannot be further merged or combined into a single analytic step. In addition, because of using visualization functions to provide users with new insight before each intermediated analytic step, we can model the multi-operator process as independent steps, where the user defined operator distribution set based on uncertain predicates' probability are independent. By updating the graph probability distribution set, $\Omega_G$, after each single operation, we keep tack of all the necessary probability information from all past uncertain querying steps within the probability distribution set as a self-contained graph attribute.Based on such construction, the final output graph probability will only depend on its input graph's probability from one step before and the user defined operator distributions from the current analytic step. If all output graphs result from the same set of analytic steps, we can rank all output structures based on their output probabilities, which describe how well the given graphs fit the user's target structure.

Figure 3.8 shows a example of composition of operator process that involves two separate steps; an aggregation step, $A1$, and a selection step, $S1$. The probability distribution set of $A1$ and $S1$ based on their uncertain predicates are independent. According to the construction of the composition process, all graph probabilities in the final output set $D''$ are independent from graph probabilities in $D$ given the $D'$s probability information. The output graph probability will only depend the graph probability distribution set from its input graph $D'$ and user imposed probabilities based on the uncertain operator $S1$.

It is worth to mentioning that a single operator step is a special case of the composition process which contains only one analytic step. In addition, the operator

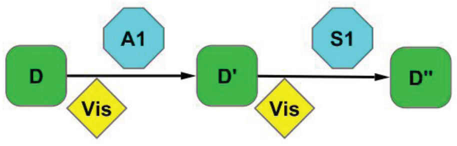


Figure 3.8: Example of a workflow of an analysis with usage of visualization function.

based on an exact predicate can be treated as a special case of the analytic process based on an uncertain predicate with a probability distribution for the exact predicate set to $\Omega_{predicate} = (1)$. Since there is no additional information on the probability list for any initial input graphs, we assume each of the initial input graphs to have a probability list of $\{\{1\}, \{1\}\}$.

### 3.6.2 Social Networking Application

In order to illustrate the usability of the uncertain predicate framework in a real world setting, we employ the Flitter network as listed in *VAST* (2009),a synthetical social computing tool such as Twitter, in this study. Moreover, we demonstrate the effectiveness in reducing users' workload of uncertain querying by dividing the complex uncertain predicate into smaller simple uncertain predicates. Finally, we compare the final outputs between our method to other existing methods.

The Flitter network contains over 6000 user IDs which are represented by nodes in the network. For each user ID, we have information on its user name, city location and size of the given location. There are 29876 edges in the network. An edge exists between a pair of nodes (or two users), if there is at least one communication between them. The goal of this study is to identify a possible criminal network based on a list of given criteria. The suspect criminal network should have 6 members in total with one employee, three handlers,one middleman and one group leader. Within their circle, they should form a network structure listed as shown 3.9. The user is certain about more connections among all members in the network except the relationship

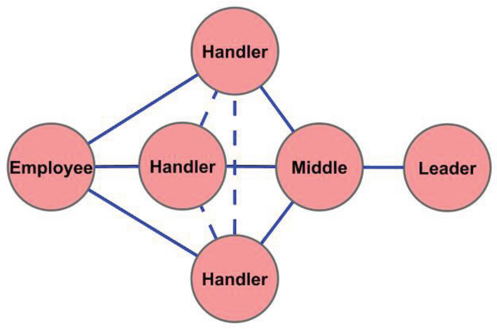


Figure 3.9: This figure shows that the suspect criminal network structure for the Flitter study. Dashlines stand for uncertain connections.

Table 3.1: Summarization of given assumptions for Flitter network study. (* stands for the uncertain requirement)

| Name | Assumptions |
|---|---|
| Employee | A user with around 40 connections.* |
| Handler (3) | Direct neighbor of employee.<br>A user with 30 to 40 connections.<br>Unknown connectivity among three handlers.* |
| Middleman | Common neighbor of three handlers.<br>A user with around 4 connections.* |
| The group leader | Direct neighbor of the middleman.<br>A user with over 100 connections. |

among the three handlers, therefore, we use a dashline to denote such the uncertain structures. In addition to the structural requirement for the criminal network, there is a list of attribute assumptions for each of the criminal members. The detailed assumptions are listed in Table 3.6.2. Based on the output network, we further study the location distribution of the connections of all members in the network of suspects. Figure 3.10 shows an overview of the given network in Cytoscape. At this stage, it is hard to effectively visualize any useful information for the force directed layout.

There are two steps in the analysis of this problem such as in Figure 3.11. First, we need select the suspected criminal network based on a uncertain query and visualization function, then we can aggregate the network and study its neighbors' distribution. Therefore, we can divide the analytic process into two independent components due to the usage of visualization function. We use the existing Cytoscape capability to

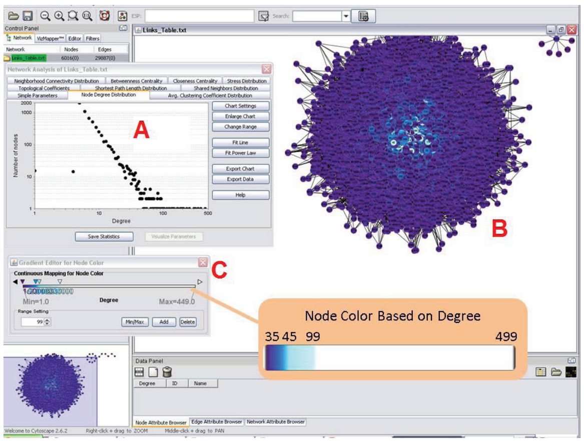


Figure 3.10: Overview of the Flitter network.

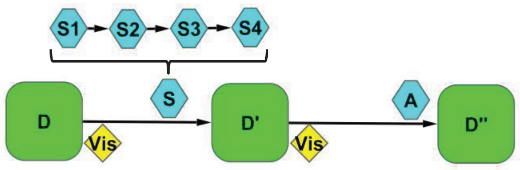


Figure 3.11: Workflow of Flitter study analytic process.

visualize the data throughout the entire analysis.

By construction, since there exists no pervious knowledge about the input Flitter network, we can assume that the initial input graph has a probability list given by $\{\{1\},\{1\}\}$. we The first independent component involves a complex structural selection operator, which contains both attribute and structural uncertainty. Due to the decomposition property of the selection operator, we can further break the selection step into a few conditional independent components. We begin our first analysis step by selecting all nodes with degree $\approx 40$ and their neighbors up to the $7^{th}$ consecutive steps, since we are interested in uncovering a network of size six and its directed neighbors. In this step, we use an element selection operator, where we select qualified nodes and use the labeling function to add an node attribute, $X_{v.label_1}$, for the target node with an $\approx 40$ condition. With the user defined probability set $\Omega_{\alpha_1^*}= (\frac{1}{3}, \frac{1}{2}, 1, \frac{1}{2}, \frac{1}{3})$, we transform this uncertain predicate into a set of exact predicates with degree equal to 38,39, 40, 41 and 42. Since the given assumption indicates the suspected employee should have around 40 connections, so the user defines higher probabilities to predicates with degree requirements close to the given assumption, and probability zero to any predicates with 3 degrees away from the given assumption. As listed before, $\Omega_{\alpha_1^*}$ is only a probability set. Each element of the defined probability set follows an independent distribution which describes how well the exact predicate structure captures the given target structure. After the first element selection step, there are 26 output graphs with different output probabilities. By requiring user input probabilities in the predicate step instead of the output step, we significantly limit the user's workload from attaching probability scores to all 26 outputs to only define 5 different predicate requirements.

Based on the selected outputs and their updated probability lists, we are going to select output graphs that contain 3 directed neighboring nodes of employees and with degree between 30 to 40. Since the user believes that three handlers work in parallel on separate cases, there are less likely to have connections than not connection at all among them. Based on this assumption, the exact predicate set contain only two exact predicates. They are $\alpha_{2.1}$ with no connection ($P(\alpha_{2.1}) = 1$), $\alpha_{2.2}$ with only one connection($P(\alpha_{2.2}) = \frac{1}{2}$). $\Omega_{\alpha_2^*}$=($P(\alpha_{2.1})$, $P(\alpha_{2.2})$) contains only conditional probabilities, for example,$P(\alpha_{2.1})$ is the probability of no connections among all three handlers given the selected employee from the previous step is a correct suspect. Because the user applies the set selection operator here, one selects the entire qualified graph instead of only the qualified elements. Moreover, we also use labeling function to create a new node attribute for the selected handlers. At the end of this step, the size of the output list is further reduced to 9 outputs and the probability attached to each output is updated based on the input probability distributions and the current operator distribution.

The next step is to select the middleman who is the common neighbor of 3 handlers with degree $\approx 4$. Similar to the first selection step, the exact predicate set contains predicates that have the same structural requirement, but different degrees ranged from 2 to 6 with probability set $\Omega_{\alpha_3^*}$. In this step, the user picks the probability set $\Omega_{\alpha_3^*}$ = (1,1 ,1, $\frac{1}{2}$, $\frac{1}{3}$), since one believes that the middle man is more likely to be less active than a highly connected one. Once again, the user also uses set selection to obtain outputs. The $\Omega_{\alpha_3^*}$ set is conditional independent from $\Omega_{\alpha_1^*}$ given $\Omega_{\alpha_2^*}$. By adopting the conditional independent assumption amongst all consecutive analytic steps, our framework effectively decreases the user's workload in defining joint distributions for the complex uncertain predicate.

The final step of the first independent component is to discover the leader of the criminal network. Based on the given assumption, the group leader should be a highly connected node with degree over 100. This step is a set selection operator based on an exact predicate structure. The exact predicate query is treated as a special case of the uncertainty problem in the analytic process with one exact predicate structure in the set with probability one, therefore the operator distribution set for each output in this step should be [1].

After all selection steps, only 2 outputs in Figure 3.12 are left with different output probabilities based on the graph probability distribution sets as follows:

$$[1] \otimes [0,0,1,0,0] \otimes [1,0] \otimes [0,0,0,\frac{1}{2},0]$$
$$[1] \otimes [0,0,1,0,0] \otimes [1,0] \otimes [0,0,0,0,\frac{1}{3}]$$

We rank all outputs based on their probabilities. The user can use visualization function to further eliminate the size of output list if needed. Since there are only two output graphs left in this study, we keep all of them for the next step.

**Selection Step:**

1. Element selection

   $\alpha_1^*$={$\{v_1, \ldots, v_7\}$,$\{e_{12}, \ldots, e_{67}\}$, $X_{v1.degree} \approx 40$}

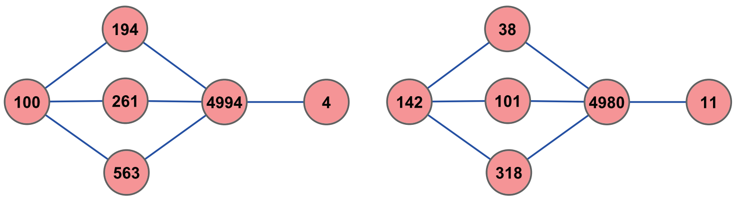


Figure 3.12: Two output graphs from the first selection component. Number in each node is the user ID. The output probability for the upper graph is $\frac{1}{2}$ and the output probability for the lower graph is $\frac{1}{3}$, because of the degree differences between two middleman, user 4994 and 4980.

Exact predicate set for $\alpha_1^*$:

- $\alpha_{1.1}$=$\{\ldots, X_{v1.degree} = 38\}$
- $\ldots$
- $\alpha_{1.5}$=$\{\ldots, X_{v1.degree} = 42\}$

with probability set $\Omega_{\alpha_1^*}$=$(\frac{1}{3}, \frac{1}{2}$,1,$\frac{1}{2}, \frac{1}{3})$.

2. Set selection

   $\alpha_2^*$=$\{\{v_1, \ldots, v_4\}$,

   $\{e_{12}, e_{13}, e_{14}, \approx e_{23}, \approx e_{34}\}$,

   $X_{v1.label_1}$ =”Y”,$X_{v2-4.degree} \in [30, 40]\}$

   Exact predicate set for $\alpha_2^*$:

   - $\alpha_{2.1}$=$\{\{v_1, \ldots, v_4\}$,$\{e_{12}, e_{13}$,$e_{14}\}$,$\{!e_{23}$,$!e_{34}\}, \ldots\}$
   - $\alpha_{2.2}$=$\{\{v_1, \ldots, v_4\}$,$\{e_{12}, e_{13}$,$e_{14}, e_{23}\}$,$!e_{34}, \ldots\}$

   with $\Omega_{\alpha_2^*}$=$(1, \frac{1}{2})$.

3. Set selection

   $\alpha_3^*$=$\{\{v_1, \ldots, v_4\}$ ,$\{e_{14}$,$e_{24}, e_{34}\}$,

$X_{v1-3.label_2}$ = "$Y$", $X_{v4.degree} \approx 4\}$

Exact predicate set for $\alpha_3^*$:

- $\alpha_{3.1}$=$\{\ldots$,$X_{v4.degree} = 1\}$
- $\ldots$
- $\alpha_{3.6}$=$\{\ldots$,$X_{v4.degree} = 6\}$

with $\Omega_{\alpha_3^*}$=$(1, 1, 1, \frac{1}{2}, \frac{1}{3})$.

4. Set selection

   $\alpha_4$=$\{\{v_1, v_2\}$, $e_{12}$, $X_{v1.label_3}$="$Y$",

   $X_{v2.degree} > 100\}$.

For the second part of the analysis, where one is interested to study the location distributions of the suspect criminal network, the user can visualize directly of the location distribution of neighbors for each of the members. In order to have a better understanding of the overall distribution of all neighbors, we use element aggregation operator to aggregate all members of the network together to a super node and call a summary function to compute the location distribution of neighbors of the super node. Since this step is constructed based on a exact predicate, the witness distribution stay unchanged from the network structure to the super nodes. We compute a location distribution of all neighbors.

## 3.7 Conclusions

In this chapter, we presented a probability model for uncertainty in queries. We incorporate this model into a visual analytic algebra. We present multiple examples on how it can be used. We also show the probability can be propagated in a workflow of uncertain predicates. We would like to test this system with real users and situations to explore its benefits. There are still many areas of uncertainty in graphs to explore. The next step would be to incorporate uncertain predicates to an uncertain graph model. Another challenging problem is how to maintain the lineage of the different uncertainty values created in a workflow.

CHAPTER IV

# Visualizing High Dimensional Data with Network Constraints

## 4.1 Introduction

In recent years, effective and perceptually meaningful visualizations of high dimensional data is a topic that has received a lot of attention from the visualization community. This is due to the abundance of such data sets in various application. For example, in many biological settings, researchers routinely collect expression levels for functional related genes (and/or proteins, metabolites) for a large number of patients that belong to different groups (e.g. healthy, early disease stage, late disease stage). Another example comes from marketing research, where data about users preferences regarding specific products have been collected together with background information (e.g. demographic characteristics, educational level and socio-economic status)and social relations among users. It can be seen that the structural information can be encoded in the form of a graph, with nodes corresponding to patients or the product users and edges capturing their clinical similarity or social relationships, respectively.

Techniques for visualing high-dimensional data, such as principal components, multidimensional scaling and linear discriminant analysis (*Jolliffe* (2002); *Borg and*

*Groenen* (1997); *McLachlan* (2004)) provide summaries of different aspects of the data that lead to useful visual exploration. However, the additional information regarding relationships encoded in the graph is not utilized. Similarly, a vast body of literature has emerged on techniques for network visualization such as *Viau et al.* (2010); *Muelder and Ma* (2008). Once again, node characteristics information (in the form of data sets as explained above) is not received adequate attention in the network visualization process. Hence, most of existing visualization techniques either focus on the high dimensional data or on the network relationships, but do not seamlessly integrate both in visual displays. As pointed out in *Grammel et al.* (2010), this can be particularly problematic for visualization novices, because inadequate (or incomplete) visual mappings can be misleading.

In this paper, we introduce a dimension reduction technique that aids in the visualization of high-dimensional data, but in addition incorporates the structural information available in the form of a network constraint. It is based on a penalized decomposition of the data matrix, with the penalty encoding the network constraint. This leads to a visual display that effectively contains both sources of information. We present a fast algorithm to obtain the decomposition and show that it is easy to implement in existing systems. Specifically, the matrix decomposition algorithm is implemented in R (*Hornik and Gebhardt* (1998)), a free statistical computing platform and network visualization in Cytoscape (*Collaboration* (2006)), a popular visualization for biological data. The remainder of the paper is organized as follows: in Section 2, we briefly present related work. In Section 3, the proposed approach is introduced and algorithmic issues discussed. Application to real data are given in Section 4 and 6. Last, some concluding remarks are drawn in Section 7.

## 4.2 Related Work

Dimension reduction techniques lead to useful visual mappings that capture important features in high-dimensional data. One of the most popular ones, is principal components analysis (PCA) (*Jolliffe* (2002)) that uses linear combinations of the features (variables) in the data to construct a low (usually two or three dimensional) subspace that is easy to visualize. The coefficients of the linear combination are chosen to maximize data variance. In the presence of a priori assigned groups for observations in the data set, linear discriminant analysis (LDA) (*McLachlan* (2004)) can also prove to be useful, since it projects the data onto a low dimensional subspace that maximally separates the groups to emphasize their differences in the resulting visual mapping. In many instances, these techniques are enhanced and combined with other data mining techniques, such as clustering to further filter the results (*Choo et al.* (2009); *Oesterling et al.* (2010)).

For data sets with a few dozen features, popular visualization techniques such as parallel coordinates plots and scatterplot matrices can also prove some insightful information. In paper(*Elmqvist et al.* (2008b)), a new interactive strategy is presented for exploring multidimensional data using scatterplot matrices. In *Yuan et al.* (2009), more effective ways of displaying multidimensional data are introduced. In paper *Ma* (2003) visual strategies for looking at time varying data are discussed. Some visualization systems (e.g. MulteeSum *Miriah Meyer and Pfister* (2010) and FlowVizMenu *Viau et al.* (2010)) link a number of visual displays that capture different aspects of the data, which together along with the usage of differential coloring and plotting symbols enhance the visual exploration.

Finally, there has been work on building an algebraic framework for supporting visual analytics (*Shaverdian et al.* (2009a); *Moustafa et al.* (2011)), thus making visualization a component of a more complex data mining and exploration pipeline.

## 4.3 Network Penalized Matrix Decomposition Approach for Dimension Reduction Method

### 4.3.1 Network Penalized Matrix Decomposition

Before we formally introduce our technique, let's briefly go over the problem setting first. We assume that data have been collected in a matrix $X$ with $n$ rows corresponding to samples (observations) and $p$ columns corresponding to features (variables). Further, it is assumed that observed relationships among the samples and the variables are encoded in graphs $G_s$ and $G_f$, respectively. These graphs can be mathematically represented by their adjacency matrices $A_s$ of size $n \times n$ and $A_f$ of size $p \times p$, with non-negative entries that represent the strength of their relationships between samples (and/or features).

The proposed technique builds upon the singular value decomposition of the data matrix $X$. Formally, $X$ can be written as $X = U\Lambda V'$, where $U$ is a $n \times p$ matrix containing the left-singular vectors, $\Lambda$ is a $p \times p$ diagonal matrix containing the singular values and $V$ is a $p \times p$ matrix containing the right singular vectors, with $U'U = I$, $V'V = I$. It is worth mention that the columns $u_i$ of $U$ corresponds to the eigenvectors of the matrix $X'X$, while the columns $v_j$ of $V$ corresponds to the eigenvectors of $X'X$. The Housholder-Young theorem (*Gene H. Golub* (1996)) states that if we keep the $k$ left and right singular vectors $(U^k, V^k)$ which are corresponding to the largest $k$ singular values $(\Lambda^k)$, then the resulting matrix $\tilde{X} = U^k\Lambda^kV^k$ gives the best $k-$ rank approximation in the Frobenius norm (i.e. $\min_{\tilde{X}} ||X - \tilde{X}||_F^2$).

Our approach incorporates the network information through penalties on the left and right singular vectors. Specifically, we want to minimize the following objective function

$$\min_{U,\Lambda,V} \parallel X - U\Lambda V' \parallel_F^2 + \delta_1 P_1(U) + \delta_2 P_2(V)$$

for some tuning parameters $\delta_1$ and $\delta_2$. The penalty function of observations takes the form $P(U) = U'L_sU$, where $L_s = I - D_s^{-1/2}A_sD_s^{-1/2}$ is the normalized Laplacian matrix of the graph $G_s$ and $D_s$ contains the row sums of the adjacency matrix $A_s$. By the construction of $L_s$ matrix, we can rewrite the constraint as $u'L_su = \sum_{i,j}(u_i - u_j)^2 \frac{a_{ij}}{\sum_{ij} a_{ij}}$ for a rank-1 estimation of $u$ with $a_{ij} \in A_s$ is the $ij^{th}$ term in the adjacency matrix of observations. It is easy to see that the network constraint pushes connected low-degree nodes in the graph closer together, when projected to a Euclidean space. The intuition behind our objective function is illustrated in the later section where connected observations are placed closely together in the projected subspace obtained from the network penalized singular value decomposition. Further, in the presence of very high-dimensional data, one may want to create a sparse solution for the right singular vectors by considering a constraint of the form $P(V) = ||V||_{\ell_1}$ ,$||\cdot||_1$ denotes the $\ell_1$ norm. The $\ell_1$ norm constraint sets many of elements in $V$ equal to 0, thus keep only the most "significant" features in the reconstructed $\tilde{X}$ for the interpretation propose. It is worth to point out that we can always left out the $\ell_1$ norm from the penalty constraint functions for pure visualization problems. In cased of the absence of additional network information. the Laplacian matrix can be set to zero matrix. Then our network penalized SVD will only have $\ell_1$ norm penalty. The solution of the given objective function will be the solution for sparse PCA.

After some simple algebraic manipulations, we can find the rank-one estimation for the objective function for a fixed $\lambda \in \Lambda$ by solving the following problem

$$\begin{aligned}
\max_{u,v} \quad & u'Xv \\
\text{Subject to } u'L_su \quad & \leq \quad c_1 \\
v'L_fv \quad & \leq \quad c_2 \\
\text{or } ||v||_1 \quad & \leq \quad c_2 \\
u'u = 1 \quad & , \quad v'v = 1
\end{aligned}$$

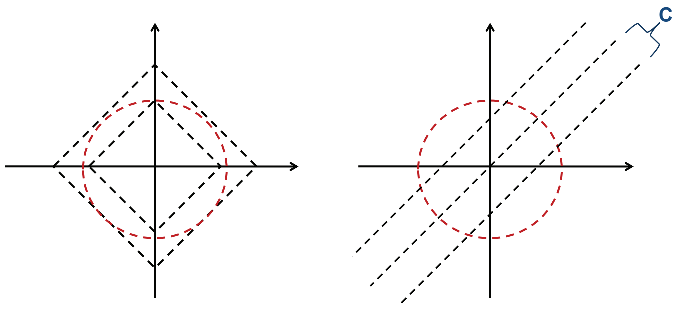


Figure 4.1: A two-dimensional visual representation of penalty functions.

where $u$ and $v$ are $n$ and $p$ dimensional vectors.

In Figure 4.1 shows an illustration of the penalties. Specifically, the left panel depicts the $\|v\|_1 \leq c_2$ and $v'v = 1$ constraints, while the right panel depicts the $u'L_s u \leq c_1$ and normalization constraints. It can be seen that when $c_1 \geq \sqrt{p}$, only the normalization constraint (red) is active, while $1 \leq c_1 \leq \sqrt{p}$ the active constraint becomes a diamond shape with four cutoff ends. Dependent on the direction of $u'X$, the estimated $v$ can be either on one of the constraints or their intersections. The network constraint is controlled by the constant constraint value and the given adjacency matrix $A_s$. When the weight between two connected components goes to infinite, the constraint band will force the connected components to have the same new coordinates in the low dimensional space.

### 4.3.2 Estimation and Algorithm

The optimization problem under consideration is a biconvex one ( *Witten et al.* (2009)). We can employ a block-relaxation algorithm (*de Leeuw and Michailidis* (2000) ) to find the optimal $u$ and $v$. First, we estimate $u$ for a input $v$ value based on Newton's method for nonlinear system of equation. Then, we solve for $v$ for estimated $u$. Since we use $\ell_1$ norm in the penalty function for sparsity, the constraint function is not differentiable; therefore, we need to estimate $v$ in a case by case fashion, which is similar to the method listed in *Tibshirani* (1996). Finally, the value of $\lambda$ (the singular value) is obtained by setting $\hat{\lambda} = \hat{u}X\hat{v}$.

```
Algorithm 1: Rank-1 estimation
  Input: $X$, initial $v_0$ and constant constraint values $c$
  Iterate until convergence:
      • Estimate $u$ for fixed $v$
        $min_u - u'X\hat{v}$, subject to $u'L_s u \leq c_1$ and $u'u = 1$
      • Solve
        $min_v - \hat{u}'Xv$, subject to $v'L_f v \leq c_2$ or $\| v \|_1 \leq c_2$ and $v'v = 1$
  Compute $\lambda = \hat{u}'X\hat{v}$
  Output: $\hat{u}$, $\hat{v}$ and $\hat{\lambda}$
```

We can solve the rank-k approximation problem, by repeatedly applying the rank-1 algorithm 1 on the residual matrix $X - \tilde{X}$, where $\tilde{X} = \hat{\lambda}\hat{u}\hat{v}'$ (see algorithm 2). Due to the presence of the additional constraints, $u_1, \cdots, u_k$ are not orthogonal (and similarly for the $v$'s). Note that in the absence of network constraints, we obtain the sparse principal components solution. In many cases, when we have an equal degree of confidence in both the high-dimensional data and the network constraints, one can fix a constant $c$, with $c_1 = c\sqrt{n}$ and $c_2 = c\sqrt{p}$, to obtain similar amounts of penalization for $u$ and $v$. In principle, the constants $c_1$ and $c_2$ can be chosen either according to the degree of confidence on $X$ and the constraints, or by data driven; e.g. through cross-validation as applied to subsets of the data. If one wants to select constant constraint values based on cross- validation method, he or she can randomly select the $ij^{th}$ term of data matrix and treat it as the test set. The optimal constraint values are select based on minimizing predication errors. The similar methods is also adopted by *Olga Troyanskaya et al.* (2001)'s work.

**Algorithm 2:** Compute rank-k approximation

```
Input: $X$
For $i \in 1, 2, 3 \cdots, k$
    • Compute rank-1 approximation: $\hat{u}_i$, $\hat{v}_i$ and $\hat{\lambda}_i$ for $X^i$
    • Find data matrix for rank $i+1$ estimation: $X^{i+1} = X^i - \hat{\lambda}_i \hat{u}_i \hat{v}_i{}'$.
Output: $\{\hat{u}_1, \hat{u}_2, \cdots, \hat{u}_k\}$, $\{\hat{v}_1, \cdots, \hat{v}_k\}$ and $\{\hat{\lambda}_1, \cdots, \hat{\lambda}_k\}$
```

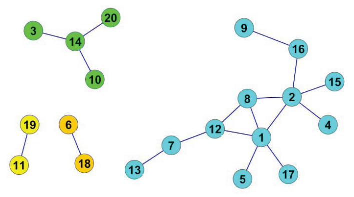


Figure 4.2: Network constraint of the synthetic data set. Color of the node represents its membership to a connected subgraph.

### 4.3.3 Illustration of the Network Penalized Singular Value Decomposition

In order to demonstrate the effectiveness of the network penalized singular value decomposition (NPSVD) approach in finding a suitable low-dimensional projection of multidimensional data that incorporates a network constraint, we compare the 2-dimension visual representations from different dimension reduction methods on a synthetic dataset. We employ only a limited numbers of observations for this study to obtain a clear picture of the connectivity imposed by the network constraint on the sample. In our example, we have a total of 20 observations and 10 features. Each observation is generated according to a multivariate distribution with diagonal covariance matrix. The network constraint is constructed at random and shown in Figure 4.2. In order to visualize the results clearly, we assign the same color to all observations within the same connected component.

The standard display of the output obtained from principal component analysis that utilizes only the $X$ data, is shown in the top left panel of Figure 4.3.The first two principal components capture about 40% of the total variance in the data. The

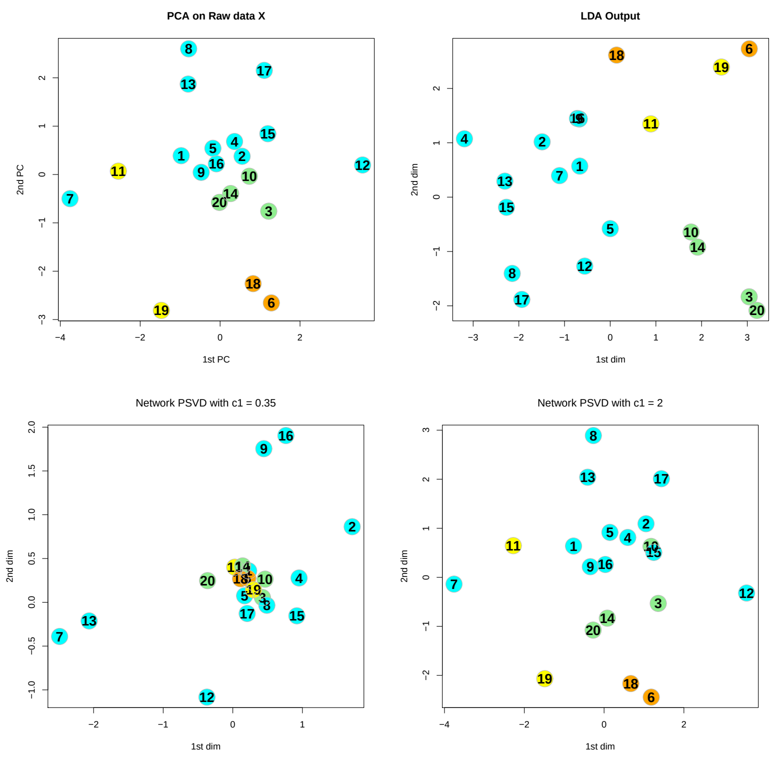


Figure 4.3: Principal component analysis output for the synthetic data set (Top Left). Linear discriminant analysis output for the synthetic data set(Top Right) with overlapping nodes are 9 and 16. NPSVD outputs with different network constraints (Bottom).

observed clustering patterns in the center of the plot are dueto similarities among those observations in $X$. Without cosidering any connectivity information, the connect observation are drafting apart in the PCA output, for example, nodes $13, 7$ and 12. In many applications, such as marketing study, freiendship which often can be represented as connected nodes in the social network, will influence consumer's preferences; therefore we would like to place connected nodes together in the low dimensional visual representations. In this case, PCA provides a misleading image of the entire data information.

A different view of the data is obtained by employing linear discriminant analysis, where we treat the connected components in the network as class labels (for a total of four groups). The resulting two-dimensional representation shown in the top right panel of Figure 4.3 is driven by the grouping assignment, since the technique is designed to maximize the between group separations and minimize the within group distances; therefore, sometimes the low dimensional space is constructed based only on a few original variables which provide a good group separations. It is easy to see there is a clear separation among different colored nodes in Figure 4.3, however, this image is constructed only based on two out off ten original features. So it is not very informative in term of capturing the high dimensional data information. In addition, by transforming the complex network information into group labels, we also suffer from information loss.

In this example, we only impose a network constraint on the samples and leave the features free. The resulting visual display of both the data and the network information is given in bottom of Figure 4.3. On the bottom left panel, the constant constraint value $c_1$ is smaller and hence the network constraint becomes more binding, while in the bottom right panel the value $c_1$ is fairly large and the display resembles PCA outputs. Specifically, the connected observations, nodes $13, 7$ and 12,come close together as expected, and the same happens for all other tightly connected observations. In the extreme case, for a very small value of $c_1$ the plot will only have four points corresponding to the centers of the four connected components in the network. This shows that one can obtain a succession of views of the data, with one extreme being a visual representation of the information contained in the data matrix $X$ and another extreme displaying the clustering structure captured by the network. Tuning the $c_1$ constraint gives views that emphasize either the data $X$ or the constraint $G_s$. A similar result holds for constraints imposed on the features through $G_f$.

## 4.4 Movie Rating Application

We employ the NPSVD to a data set of movie ratings obtained from the IMDB database. For each movie we have information about its title, year it was released,

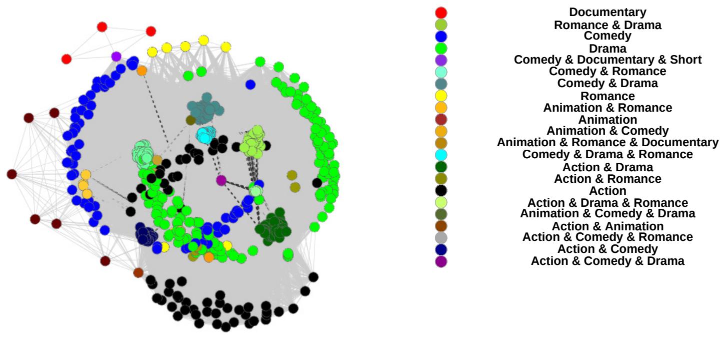


Figure 4.4: Movie network contains 558 movies (nodes) from the year of 1916 to 2005. The color of nodes represents as type of each movie. Two movies are connected, if they share at least one common category.

length, budget, ratings and total number of votes received by raters. We examine 558 movies that can be categorized to Short Feature, Romance, Drama, Documentary, Comedy, Animation and Action. Each movie can be assigned to more than one categories. For example, Toy Story (I) is assigned to both the animation and comedy categories, Braveheart is assigned to the action and drama categories, while Crash is assigned to only the drama category. In total, we can assign all the 558 in 22 non-overlapping categories. The detailed distribution of movie types is given in the Appendix. The network constraint is constructed based on the original seven categories. Specifically, two movies are connected if they share at least one category. For example, there exists no connection between Toy story (I) and Crash since they share no common movie categories, whereas there exists an edge between Crash and Braveheart since both are categorized as drama. The network is depicted in Figure 4.4, using a spring-embedding layout algorithm that utilizes edge weights. The edge weights are constructed based on number of the common categories. To enhance the visualization of the network constraint we also assigned different colors to the 22 non-overlapping categories we created. As expected, the network captures to a large extent the similarities between the movies based on their genre. A set of network statistics that describes the characteristics of the network are listed in table 4.4.

Table 4.1: Network Summary

| Network Statistics | Quality |
|---|---|
| Nodes | 558 |
| Edges | 90138 |
| Average Transitivity | 0.824 |
| Range of Node Degree | 3–539 |
| Range of Edge Weight | 0.143–0.428 |

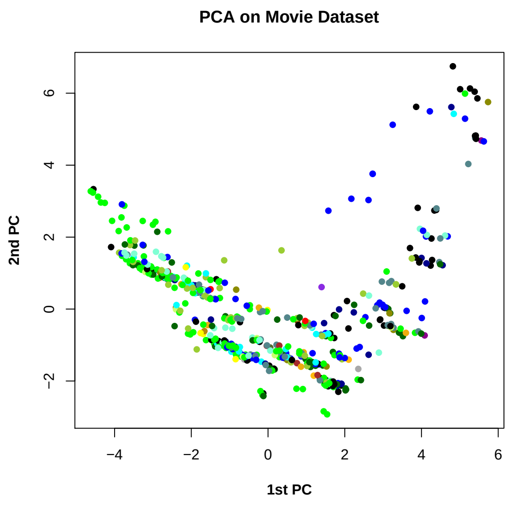


Figure 4.5: Two dimensional PCA projection of movie ratings

How does the view of the movies change when user ratings are employed? We start by showing the information contained in the ratings using principal components analysis (see Figure 4.5) and also adding the category information through coloring. It can be seen that to a large extent users that rated high drama movies, rated low comedy ones, since the drama movies (green) are mostly clustered on the left side of the plot, while comedy movies on the right side. Further, there are movies with similar ratings (overlapping in the plot) that belong to different genres. This should be expected, since a good movie may receive good ratings irrespective of its genre, and similarly for a bad one.

The visual display obtained from NPSV is given in Figure 4.6. The basic pattern observed exhibits similarities with that obtained from principal components analysis. However, the obtained solution exhibits a layering property; namely, romance and

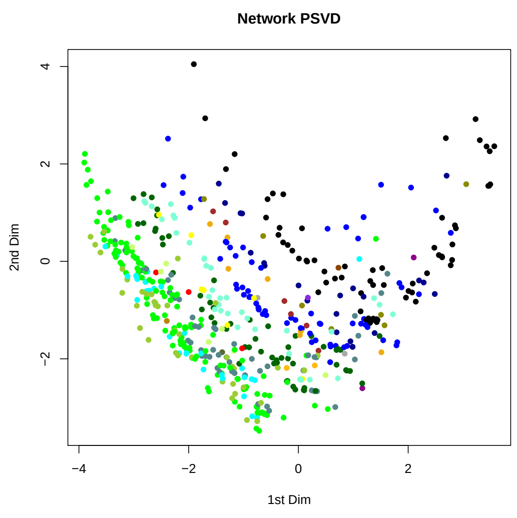
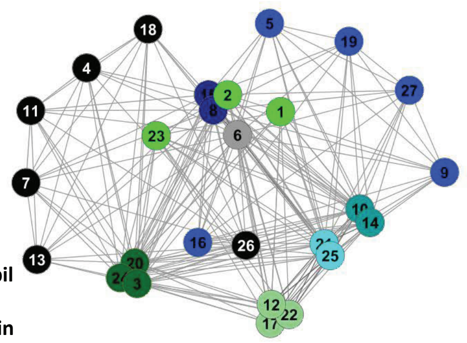


Figure 4.6: Two dimensional network PSVD display of movie ratings



1. Apt pupil
2. Crash
3. Hard rain
4. Deep rising
5. Blues brothers 2000
6. Around the world in 80 days
7. The replacement killers
8. Bad company
9. The big Lebowski
10. Primary colors
11. Lost in space
12. City of angels
13. Firestorm
14. The Truman show
15. Man of the house
16. Chairman of the board
17. The horse whisperer
18. Fall
19. Sleepover
20. The man in the iron mask
21. The object of my affection
22. The English patient
23. Fargo
24. Ransom
25. Tin cup
26. Mission: impossible
27. Happy Gilmore

Figure 4.7: Two dimensional PCA projection of a set of movie ratings

drama movies (olive green) are close to the drama ones (green), followed by action and drama movies (dark green) and so forth. Hence, the proposed technique through the network constraint introduces a clustering effect on the movie ratings, while still preserving the information available in the ratings.

In order to get a better understanding of the NPSVD solution we further examine

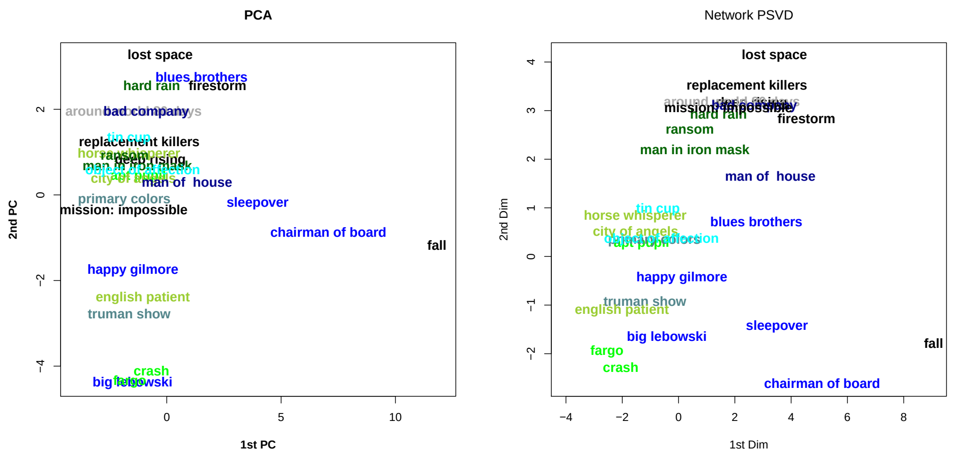


Figure 4.8: Two dimensional PCA and network PSVD projections of a set of movie ratings

twenty-seven popular movies. Figure 4.7 depicts their network connectivity and titles. The visual displays obtained using principal components analysis an NPSVD are shown in Figure 4.8. The principal components based visualization is fairly cluttered, due to the similarities in the underlying movie ratings. On the other hand, the NPSVD provides a more informative display, by separating similarly rated movies according to their genre. In this case, the network constraint effectively increases the visibility of the graph layout, while maintaining the rating information.

Overall, only the proposed network penalized matrix decomposition method effectively captures the two sources of information (data and network constraints). Obviously, unreliable connectivity patterns may lead to distorted patterns and the same can happen for highly noisy data.

## 4.5 Actor/Actress Application

Beside of dataset is constructed on each movie title, we also obtained a small actor/actress dataset from the IMDB database to visualize the rating behavior from a different aspect. There are total 18 stars in the dataset. For each star, we average rating related and box-office related information of all movies during 1990 to 1995.

Table 4.2: Number of total movies per star

| Name | Number of Movies | Name | Number of Movies | Name | Number of Movies |
|---|---|---|---|---|---|
| Kevin Bacon | 10 | Tom Cruise | 5 | Kevin Costner | 9 |
| Morgan Freeman | 7 | Tom Hanks | 7 | Dustin Hoffman | 5 |
| Jack Nicholson | 6 | Tim Robbins | 10 | John Travolta | 6 |
| Gary Oldman | 11 | Bruce Willis | 15 | Denzel Washington | 10 |
| Halle Berry | 8 | Nicole Kidman | 8 | Demi Moore | 8 |
| Julia Roberts | 9 | Meg Ryan | 9 | Meryl Streep | 6 |

For example, in the given five year period, Tom Hanks took roles in total seven movies. Some of them are more popular than others, such as Forrest Gump has both high rating and worldwide gross; while others have a fewer number of votes and low domestic gross. In order to take out some of the variation and look at the overall performance of each star in the given time period, we average movie ratings, number of votes, number of reviews and critics, revenue from the opening weekend (U.S.) and total gross (U.S. or worldwide) from all his or her movies. In order to measure each star's productivity, we also obtain numbers of total movies one made during the time period. The detail information on their total number of movies is listed in table 4.5. Based on this dataset, they made around eight movies during the five-year period on average. Bruce Willis and Denzel Washington are the most active stars in the dataset. For most of well established stars, such as Dustin Hoffman and Jack Nicholson, they are tended to select their movies more carefully. Among all seven observed variables, all variables related to ratings (average number of votes, reviews and critics, average ratings) are highly correlated, and all variables related to the box-office (average revenue from the first opening weekend and box-office gross) are also correlated. The strengths of correlation can be found in the scatterplot matrix, Figure 4.9.

The obtained network is motivated by Six Degrees of Kevin Bacon game (*Travers and Milgram.* (1969)). Two stars are connected, if they worked in the same movie. Edges within the network are weighted by number of their collaborations. We have mostly the famous stars in the business during the given time period. In addition,

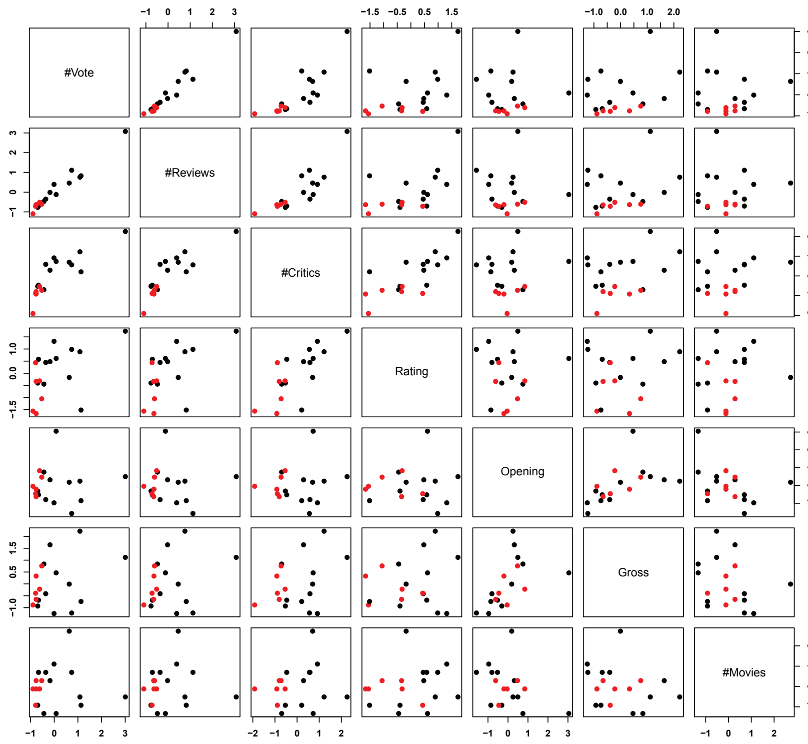


Figure 4.9: Scatterplot matrix for all seven variable in the star dataset. Black nodes correspond to actors and red nodes correspond to actress in the dataset. The seven variables are average number of votes, reviews and critics, average ratings, average revenue from the first opening weekend, average box-office gross and the total number of movie one made between 1990 to 1995.

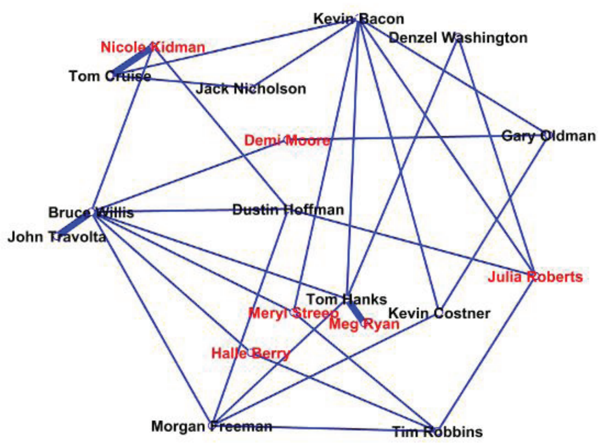
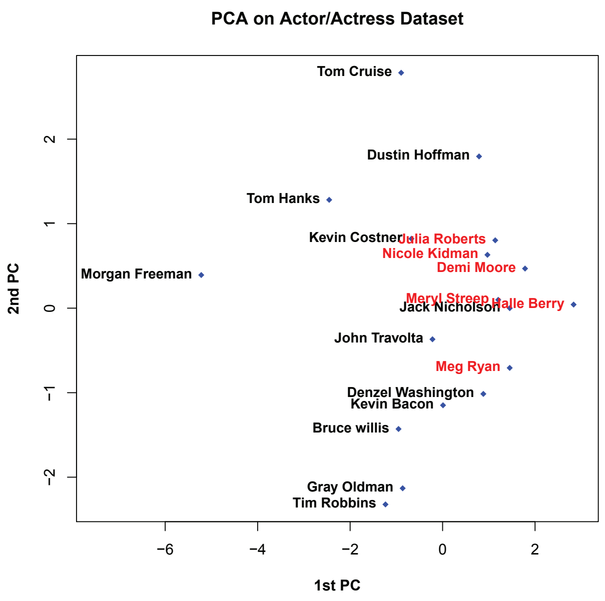


Figure 4.10: Network for all 18 stars. The wider the edge width implies more frequent collaborations.

most of movies only offer a few important roles at once; therefore, most of stars only worked once together. Only Tom Cruise / Nicole Kidman (the real life couple ), Tom Hanks / Meg Ryan (the magical movie couple which grantees the box-office) and Bruce Willis / John Travolta appeared in two movies together during the five-year period. The detail structural information can be found in Figure 4.10. The wider the edge width implies more frequent collaborations.

PCA output in Figure 4.11 is constructed based only on high dimensional data information. According to their factor loadings, the first dimension captures only the variables related to ratings, while the second dimension captures only the variables related to profit and productivity. For example, most of actresses had low ratings, while Morgan Freeman had the highest average rating during the given time period because of popular movies, such as Unforgiven, Seven and The Shawshank Redemption. Because most of Tom Cruise's films were released in cinemas worldwide, he had an outstanding average box-office gross comparing to others. On the other hand, Gary Oldman and Tim Robbins also took roles in many important films, however, their high participation in less popular and low gross domesticate films out weight their influential works in the overall average. Among the first two dimensions, it



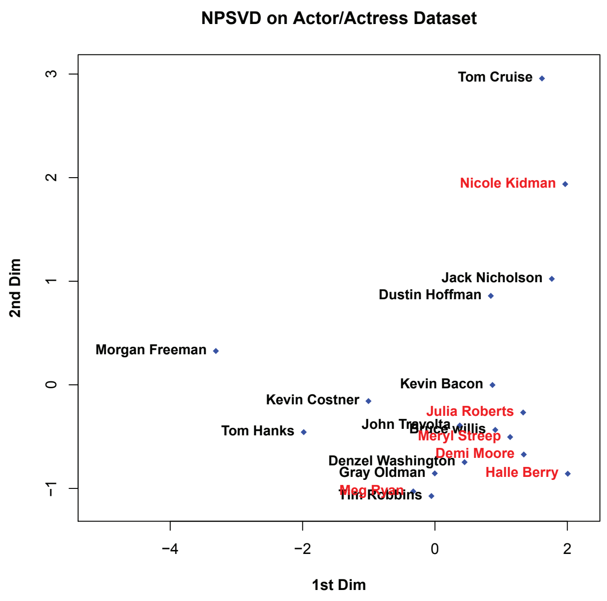


Figure 4.11: PCA output is constructed based only on the seven input variables, while Network penalized SVD output is constructed based on both characteristic and structural information

captures less than 70% of total variance in the dataset. However, there exists no information on their connections.

As can be found in both scatter plot and PCA outputs, most of actresses have lower ratings and make less revenues than their male coworkers because of limited substantial female roles in the movie industry during the early 90's. In addition, during that time period, some of actresses,such as Julia Roberts and Halle Berry, were still at the starting point of their careers, especially, Halle Berry only made one TV series before the year 1990; therefore, ratings and revenues may not be the only criteria for their overall rankings. We believe that working with other famous movie stars in the business will improve their chances of getting recognition and bring in more potentials for their future work. Based on such motivation, network penalized SVD output which contains both characteristic and structural information will provide us a more informative visual representation.

For this application, we didn't obtain any relationship information on variables, so we leave $L_f$ equal to a zero matrix. Because of the $\ell_1$ norm constraint, the first dimension of network penalized SVD captures all rating related and overall box-office information except average revenue from opening weekend and total number of produced movies. While the second dimension contains information from all variables and with a bigger weight in front of average revenue from the opening weekend. The two dimensional network PSVD output can be seen in Figure 4.11, Nicole Kidman's name stands out from other actresses because of her relationship with Tom Cruise. No only the collaboration information help actresses to have more reasonable placements, but also highly productive actor Tim Robbins is placed closer to other famous stars because of his important role in The Shawshank Redemption with Morgan Freeman. In contrast, he is placed far below the rest of the group in the PCA output since most of his movies are only released domestically.

## 4.6 Gene Expression Application

In the study of gene expression level, statistical significant of each gene expression level shifts under different experimental conditions is often assessed independently with some multiple testing corrections. As stated in *Shojaie and Michailidis* (2009), such approach can be less sensitive in detecting gene expression level changes and also lack meaningful biological interpretation. Genetics scientist (*Ideker et al.* (2001)) and Statistician (*Shojaie and Michailidis* (2009)) tried to integrate gene interaction information in different pathways, estimated correlations among genes together with the expression level dataset to identify significant expression level shifts between different experimental conditions. By incorporating additional network information, they are able to build more robust tests in finding significant and meaningful biological subnetworks and pathways.

Before applying any formal analyses, visualization is the first step of learning from the data. Since existing visualization techniques lack the ability of incorporating network information to the high dimensional data information to fully present the given information, users often have a misleading perception of the data at the beginning of analyses.

In this section, we demonstrate our network penalized SVD method on a small biological network to solve the problem. The dataset is originally presented in *Ideker et al.* (2001)'s work. The network contain 343 genes/nodes with 419 interactions/edges. In this study, we only select a subnetwork which contain two important gene pathways for a better visualization purpose. The resulting integrated physical-interaction network is listed in Figure 4.12 which contains total 25 genes and 32 interactions. Each interaction can be either a protein's physical interaction or gene transcription. The red pathway is denoted as galactose utilization in the yeast Saccharomyces cerevisiae, whereas the blue pathway is denoted as amino acid synthesis. The two pathways are connected through two genes (in yellow) with gene transcriptions. For each of the tested genes, we have total 21 perturbations (observations) in the expression level dataset.

PCA output is constructed based on only the expression level data set, Figure4.13. The color of gene names is denoted by its pathway. It is easy to see that there is no

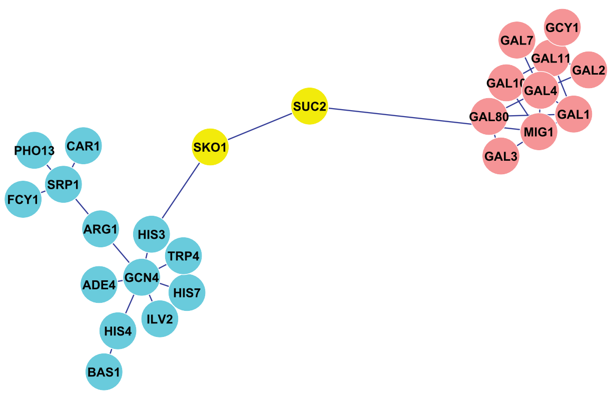


Figure 4.12: Biological network contains two gene pathways.

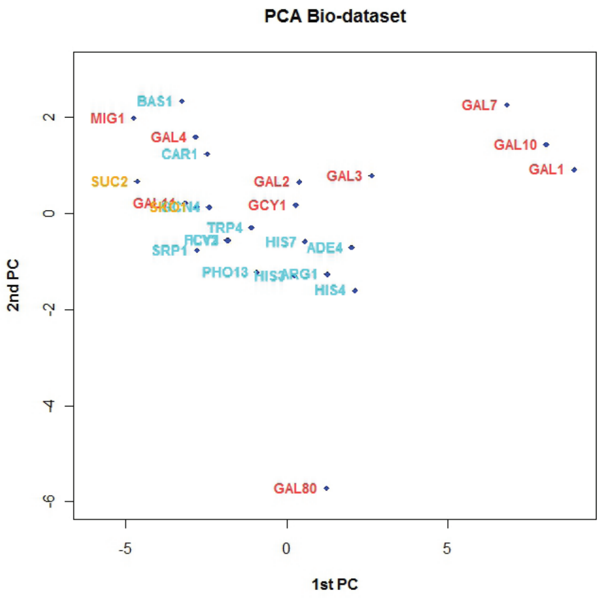


Figure 4.13: Two dimensional PCA projection of gene expression levels.

consistent patterns of gene expressions within each pathway. The connected genes may not have the similar gene expression levels.

On the other hand, we can find a better pathway separation in the network penalized SVD output, Figure 4.14. All 21 perturbations in the dataset are treated as independent observations because they are expression levels under different experimental conditions, so we have no additional structural information on the observations. Whereas, we have the structural information on features which is naturally

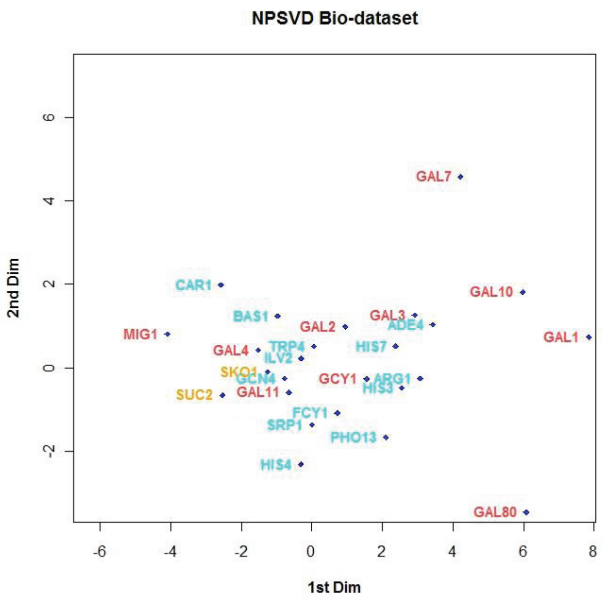


Figure 4.14: Two dimensional NPSVD projection of gene pathway dataset.

inherited from the gene pathways. Since we use the normalized Laplacian matrix which is constructed based on $G_f$ for the network constraint on features, we tend to push connected low degree genes together; therefore, all genes within the amino acid synthesis pathway (blue) is placed closer than all members in galactose utilization pathway (red). Moreover, this example also demonstrates that the natural network constraint only pull directed connected genes together with appropriated tuning parameters. Similar to the PCA output, the outliers, GAL1 ,GAL10 and GAL7, are still remained to be outliers in the network penalized SVD output.

## 4.7 Discussion and Future Work

In this study, we have introduced the network penalized singular value based matrix decomposition that creates effective visualizations of high-dimensional data sets, while incorporating constraints on the samples and/or the features. Through the utilization of network constraint, we propose a very flexible and powerful visualization technique in capturing complex relationships between observations or features.

Not only for visualizing high dimensional network data, but our method can also be used for visualizing any clustering outputs by transforming the simple class label into disconnected network components. Appropriate tuning of the constraints is critical in obtaining the most meaningful and insightful visual display of the data. For very large data sets, a data driven approach for tuning the constraints may be most appropriate. We are currently exploring more complex penalties and also how simultaneous penalties on both the samples and the features affect the visualization of the data.

# CHAPTER V

# Multi-task Learning

## 5.1 Introduction Multi-task Learning

In a standard machine learning problem, the objective is to construct a learner based on a training data set that predict responses based on future observations. In particular, responses can either be a numerical or categorical variable from the same source or population. Many models have been proposed to answer this prediction problem, for example, regression techniques for numerical responses and classification technique for categorical responses. Even in the case where the $p > n$, number of variables is bigger than number of observations, penalization approaches, such as LASSO and ridge regression, are adopted by various researchers.

In social and biological science, data is often collected from many sources. It is difficult and expensive to collect a large number of observations for each source to construct independent learners. On the other hand, due to heterogeneity within the data set, one model for entire data collection may not provide good predications for future observations. For example, in the study of Swallows population, consisting of bird observation in 2006, the migration due to major seasonal shifts makes predicting swallows' population based on the entire data set a poor approach. Another application for multi-task learning can often be found in the medical studies. When we try to study the treatment effects, data is collected from different hospitals. The demographics of patients each hospital serves may be different. A hospital in Florida may see an older patient population, on the other hand, urban hospitals may see a poorer patient population which have had less access to health care. In this special case, each month of the year or each hospital can be treated as a separate task. We can learn each task on a single, isolated basis, but would it be better for the learner to learn many similar things simultaneously? If the tasks can share what is learned, the learner may find it is easier to learn the related tasks together.

The multi-task learning considers the problem where a task will be learned with low predication error rates if it can utilize information from other related tasks during learning experiments. First, we introduce the setup of the problem. Each task is treated as a training set in a standard machine learning problem. We have $L$ tasks or samples, containing some common underlying relationships. For the $i^{th}$ task, we have observations $(x^i_1, y^i_1), \cdots, (x^i_j, y^i_j), \cdots, (x^i_{n^i}, y^i_{n^i})$ where $x^i_j = (x^i_{j1}, \cdots, x^i_{jp})$ are the predictors and $y^i_j$ is the response. In the supervised learning, the goal is to build a learning model that accurately predict new $y^i$ values. For example, to model the response $y^i$ in term of the predictors $x^i_1, \cdots, x^i_p$ alone, one may consider the linear model such as,

$$y^i{}_j = \beta^i_0 + \beta^i_1 x^i_{j1} + \cdots + \beta^i_p x^i_{jp} + \epsilon^i_j$$

where $\epsilon^i_j$ is an independent error term obeys some predefined distribution. In the above example, we learn the $i^{th}$ task by only considering the $i^{th}$ training set. Usually, we do not use information from other training sets, however, we would like to learn each single task better by building a learning model based on entire related tasks, $Y^i = f(X^1, \cdots, X^i, \cdots, X^L)$ where $X^i$ is the covariant matrix of task i.

Multi-task learning has shown its advantage in many empirical studies, however, the theoretical works on this subject is limited, see *Baxter* (2000); *Ben-David and*

*Schuller* (2003). In Baxter's theoretical work ( *Baxter* (2000)), he defines the relatedness in multi-task learning not as correlation among observations themselves, but the internal representations or the optimal hypothesis space for different tasks, where the common internal representation or the optimal hypothesis space can be captured by model structure. Details in finding the optimal common hypothesis space from a family of hypothesis spaces can be found in his work, he uses both Bayesian and information theory to demonstrate that learning multi-tasks within an environment of related tasks produce improved generalizations than learning based on a single task. By " Extending VC dimension" for family of hypothesis spaces, he derives bounds on improvements in average error rate for multi-task learning. Ben-David (*Ben-David and Schuller* (2003)) provides a more formal condition for task relatedness and tighter bounds on average error rates which capture a sub-domain of the problem. Instead of defined the related tasks share similar model structure, he defines the related tasks as samples from the similar distribution,transformations of each other. For example, different sensors collect data for the same classification problem. Each sensor has its own bias due to location. By using multi-task learning , he claims that one may eliminate complex and time consuming database integration process by finding the optimal learner based on multi-task learning method, where each database is treated as one task in the multi-task setting.

Due to the lack of theoretical justification, many authors give more restricted assumptions in relatedness. In general, researches assume that the related tasks must share some common input features. The tasks that share more input features are likely to be more related. For example, in the extreme case, we can always split one large sample into two independent samples. There exist no correlations between two tasks based on the independence assumption. Since both samples are from the same population and living in the same parameter space, we will be able to find a better learner by utilizing two tasks instead of any single training set alone.

Over the last decades, multi-task learning has appeared in many literatures within the machine learning field. Among those work, multi-task learning methods often adopt the regularization approach. For example, see *Obozinski et al.* (2006, 2008); *Pontile et al.* (2007); *Argyriou et al.* (2006); *Obozinski et al.* (2009); *Argyriou et al.* (2008); *Lounici et al.* (2009); *Chen et al.* (2010). In particular, Jordan's multi-task feature selection method (*Obozinski et al.* (2006, 2008)) is an example of the regularization approach. Jordan's method extends the $l_1$ regularization for single task estimation to the Multi-task learning setting. Since Jordan assumes that all tasks are based on the same covariant structure for a given multivariate observation, it is expected that there exists a common underlying representation shared among all classifiers. Feature selection improves the prediction accuracy and interpretation by reducing irrelevant features within a data set. A model constructed using the $l_1$ penalty by the LASSO procedure are typically sparse in the sense that only a few of the parameters are non-zero. By fitting tasks independently under the $l_1$ regularization, we might risk the possibility of losing the common underlying structure among all related tasks. Jordan's Multi-task learning method enables us to find the common feature space by penalizing the entire column of parameters at once. In his 2008 work (*Obozinski et al.* (2008)), he demonstrates the $l_1/l_2$ model carries all the regularization properties from ordinary LASSO regularization, and it gives substantial improvements in some applications.

Beside adopting LASSO's penalize strategy, many authors extend the Reproducing Kernel Hilbert space method to the multi-task learning setting. Evgeniou (*Evgeniou and Pontil* (2004)) proposed a multi-task learning method based on kernel based learning, e.g. Support Vector Machines. He tries to capture the connections among tasks by using a kernel function. Based on their 2004 work, Micchelli and Pontil extend the Reproducing Kernel Hilbert space approach into a more generalized setting. They elaborate the special kernels with more details and real world examples in their later paper (*Evgeniou and Pontil* (2004)). In their latest work (*Caponnetto et al.* (2008)), they complete vector value function kernel approach by deriving conditions that ensure the universal property of the kernel.

Many existing multi-task methods are adopting Bayesian approaches, where a prior probability model defines the relations amongst all tasks and model parameters are jointly estimated, such as *Bakker and Heskes* (2003); *Heskes* (2000). In Bakker approach, he chooses to model shared parameters through a join prior distribution which can be learned from data. In the simplest case, he defines the task depends on parameters following a prior distribution $A^i \sim N(A^i|m, \Sigma)$, so the join distribution of data becomes $P((x, y), A|\Lambda) = \prod P((x^i, y^i)|A^i, w, \sigma)P(A^i|m, \Sigma)$ with $\sigma$ as the variance of error term and $W$ as the hidden weight of tasks. The prior distribution can also be changed in order to model the clustering structures among all tasks. Furthermore, Zhang's 2008 work (*Zhang et al.* (2008)) connects the latent variable model to the multi-task learning setting following a hierarchical Bayesian approach.

It is worth pointing out that multi-task learning has also appeared in many statistics literatures other than the pure machine learning field. *Breiman and Friedman* (1997) propose the Curds & Whey method, where they solve the problem of predicting multivariate responses in a linear regression model. When there exists obvious grouping structure among tasks, hierarchial linear model approach many also used by *Raudenbush and Bryk* (2002). For example, students are grouped in schools. Hierarchical linear models capture this grouping structure by introducing indicator variables.

Beside of machine learning and statistical related area, similar problem is studied under the name of transfer learning by researchers from fields of eduction science (*J.D. et al.* (1999)), linguistics (*I.Kecskes and Papp* (2000)), psychology (*Anderson* (1995)) and human-computer interaction (*Olson and Olson* (1990)). Instead of focusing on improving prediction errors, they are interested in how individuals would transfer learning in one context to another context that shared similar characteristics more formally how "improvement in one mental function" could influence another related one.

In the work we present in this paper, a multi-task model based on generalized linear model is introduced as a possible solution to increase prediction accuracy while verifying the correct relationship amount tasks. Unlike previous work which simply assume the relationship structure amount tasks,the main feature of our model is the construction and training procedure of a weight matrix that captures relationship among tasks. The construction of weight matrix is studied rigorously on numbers of synthetic dataset. Instead of uncovering the common sparsity structures for all given tasks by applying penalty directly, we are interested in selecting and pulling only the related tasks for learning the parameters by borrowing strengths among tasks, when the feature space has a reasonable dimensionality and there exists complex relationship among tasks. We show that information on association amount tasks increases predication rate for each learner.

The rest of the paper is organized as follows. In Chapter 5.2, we introduce our new method, the generalize linear multi-task learning model. We propose an algorithm to effectively compute the solution for the model. In Chapter 5.3, we study the inference step and demonstrate the importance of constructing a weight matrix. In Chapter 5.4, a movie rating prediction problem is presented as an application. At the end, some concluding remarks are drawn in Chapter 5.5.

## 5.2 Multi-task Learning Model

In this work we deal with the multi-task learning problem with multiple tasks sharing a common optimal hypothesis space, where the common space is captured by the similarity in the observations.

We start by setting up the notation of the problem when we observe a total of $l$

samples. For the $k^{th}$ task, we have observations $(x_1^k, y_1^k), \cdots, (x_j^k, y_j^k), \cdots, (x_{n^k}^k, y_{n^k}^k)$ where $x_j^k = (x_{j1}^k, \cdots, x_{jp}^k)$ is a p-dimensional predictor and $y_j^k$ is a response. The underlying relationship between the predictors and responses within the same task can be captured by an unknown vector $\beta^k$, where *cov* $(\beta^k, \beta^j)$ may not be zero for some $k \neq j$.

We try to estimate the optimal $\beta^k$ for each task $k$, by using information obtained from related data sets. The relationship is either given based on background knowledge of data sets or pre-computed based on a similarity measure $\mathcal{R}^{l\times l}$ from available observations. The details of computing the similarity measure is given in the following section. Based on a similarity measure, $W \in \mathcal{R}^{l\times l}$, we treat the multi-task learning problem as a standard smoothing problem with weight matrix $W$. Instead of smoothing over the space defined by observations, we use the information in $W$ to smooth over the space spanned by tasks. In this setting, we maximize log likelihood function,

$$L = \sum_k^l \sum_i^l w_{ki} L^i(y^i, \beta^k x^i)$$

a summation over all tasks with weights information $W$. Since the log likelihood is constituted of independent components, maximizing the entire log likelihood function can be solved by maximizing each independent component separately.

### 5.2.1 Weight Matrix

Let $W$ denote the weight matrix containing information among all tasks. It is a standardized weight matrix with row sums equal to one. Before standardization, each element of the weight matrix $w_{ij}$ is given either as a constant weight directly or as an unknown constant based on observations. For example, the adjacency matrix of an underlying social network structure can be treated as the smoothing matrix in the study of customer's preferences, because friendship relations often have strong influence on personal preferences. Friendship often has high connectivity within groups and lower connectivity between groups, which can be defined according to different criteria; such as, socio-economic status or geographical location. Further, the weight matrix, captures such relationship, will often have block structures to capture the grouping structure within the social network. In contrast, if the weight matrix is not given directly from the background of the data set, we will need to compute it based on a similarity measure.

Because of the flexibility of the proposed model, the distance function can take many forms. The procedure of defining a distance metric for construction of the weight matrix among tasks is similar to finding the distance between pairs of observations. Since there are many existing similarity measures for observations, we can define the distance metric among tasks based on them. For example, one possible distance metric between two tasks is the average similarity measure for all pairs of observations between two samples. Based on the data sets, we can choose a similarity measure by capturing the pairwise relationship. Let's denote the similarity measure of choice as $S(x_i^s, x_j^t)$ where $x_i^s \in T^s$ and $x_j^t \in T^t$. $s$ and $t$ are indexes of different tasks; therefore, we have $w_{st} = \frac{1}{N} \sum_{x_i^s \in T_s, x_j^t \in T^t} S(x_i^s, x_j^t)$, where $N$ equals to all possible observation pairs between task $s$ and $t$. This construction will borrow strength from tasks following similar distributions.

Depend on the problem at hand, we can choose the type of similarity measures for computing $S(x_i^s, x_j^t)$. For the simplest case, we can use Euclidian distances to capture the similarity among tasks. However, when the number of tasks gets large, we would like to have a sparse weight matrix. In order to uncover the sparsity pattern, we impose some hard constraints, such as smoothing over a fixed number of neighbors, or some soft constraints, such as a kernel function with tuning parameters. The optimal tuning parameters can be obtained through cross validation on the prediction error rate. The optimal tuning parameter also provides insightful information on the true underlying relationship among all tasks. Moreover, if one believes that true parameter space can be partially uncovered by pre-fitted estimations from building individual learners, the weight matrix can also be constructed based on pre-fitted estimations.

### 5.2.2 Estimation

We utilize generalized linear models to estimate underlying parameters by maximizing the following loss function:

$$L(\beta^1, \cdot, \beta^l) = \sum_{k=1}^{l} \sum_{t}^{l} w_{kt} \sum_{i}^{n^t} [\frac{(y_i^t \gamma_i^k - b(\gamma_i^k))}{\tau^2} - c(y_i^t, \tau)] \tag{5.1}$$

Since the log likelihood is the summation of independent components, maximizing the entire log likelihood function can be solved by maximizing each independent component separately. Adopting the generalized linear model notation, the link function, $g(\mu_i^k) = x_i^k \prime \beta^k$ is consistent among all related tasks, which links together the mean of responses, $\mu_i^k$, and linear form of predictors, $x_i^k \prime \beta^k$. The $\gamma_i^k$ is a known function of expectation of $y_i^k$ and $\tau$ is the normalize constant. For example, $\gamma_i^k = \mu_i^k$ and $\tau = var(x_i^k)$ in the normal distribution case. The likelihood function for the $k^{th}$ task is a weighted sum of $l$ standard generalized linear models. Instead of assigning a weight for each observation, we weight each task based on the underlying relationships among tasks. In other words, we weight all observations the same within one task.

To estimate the optimal $\beta^k$ that maximizes the log likelihood function, we need to solve the normal equations, $\frac{\partial L(\beta^k)}{\partial \beta^k} = 0$, for the log likelihood function. Since all $k$ normal equations maybe nonlinear in parameters, we employ the iterative reweighted least squares method to solve the problem at hand. The iterative reweighted least squared method for a standard generalized linear model can be derived by using Fisher's scoring method (*WIKIPEDIA* (2011)). It takes the form $\beta^{m+1} = \beta^m + (I(\beta^m))^{-1}\frac{\partial l}{\partial \beta}|_{\beta^m}$ where $\beta^m$ is the parameter of interest at the $m^{th}$ step and $I(\beta^m)$ is the Fisher information matrix evaluated at $\beta^m$. Analogously, we can find the optimal $\beta^k$ for our multi-task generalized linear model as following.

$$\begin{aligned}
\beta^{k^{m+1}} &= \beta^{k^m} + (\sum_t w_{kt} x^{t\prime} \mathcal{Q}^t x^t)^{-1} (\sum_t w_{kt} x^{t\prime} \mathcal{Q}^t \Delta^t (y^t - \mu^t)) \qquad (5.2) \\
\mathcal{Q}^t &= \{_d [\frac{\partial^2 b(\gamma_i^t)}{\partial \gamma_i^{t2}} (\frac{\partial g(\mu_i^t)}{\partial \mu_i^t})^2]^{-1}\} \\
\Delta^t &= \{_d \frac{\partial g(\mu_i^t)}{\partial \mu_i^t}\}
\end{aligned}$$

where $\mu_i^t$ and $\gamma_i^t$ are functions of $\beta^m$. In equation 5.2, $w_{kt} \in W$ is a pre-computed weight matrix and $g(\mu_i^t)$ is the link function from the generalized model. For each iterative step $m+1$, we have $\beta^m$ as a constant vector from the previous step. Function $\mu_i^t$ and $\gamma_i^t$ are evaluated again based on constant value $\beta^m$. The main computational issues left for estimation using the multi-task learn algorithm are calculating the inverse and estimating the tuning parameter for weight matrix $W$. We are going to cover those issues in following examples.

In the simplest case, where $y_i^t$ is an independent quantitative variable following the normal distribution, the multi-task model can be simplified into a weighted regression problem. The optimal estimation for parameters $\beta^k$ for the $k^{th}$ task can be written explicitly as

$$\widehat{\beta^k} = [\sum_t^l w_{kt} \sum_i^{n^t} x_i^{t\prime} x_i^t]^{-1} [\sum_t^l w_{kt} \sum_i^{n^t} x_i^{t\prime} y_i^t] \qquad (5.3)$$

Notice that we can always pre compute $x^{k\prime} x^k_{p\times p}$ and $x^{k\prime} y^k_{p\times 1}$, so the only term we need to update is the tuning parameters in the weight matrix. For a given constant weight matrix, the multi-task method's computational complexity is same as the independent method. According to the equation 5.3, the solution always exists when the first part of the expression is invertible. It is easy to see that as long as one of $\sum_t^l w_{kt} \sum_i^{n^t} x_i^t \prime x_i^t$ with nonzero weight is a positive definite matrix, the sum will be invertible. By adopting multi-task method, we are able to construct a better learner for task with a few observation without adding too much computational cost.

In contrast, the log likelihood function for 0/1 responses can be written as equation 5.4 with $E(y_i^t) = \pi(x_i^t)$.

$$\begin{aligned} L(\beta^1, \cdots, \beta^k) &= \sum_k \sum_t^l w_{kt} \sum_i^{n^t} y_i^t log(\pi(x_i^t)) + (1 - y_i^t) log(1 - \pi(x_i^t)) \quad (5.4) \\ \pi(x_i^t) &= \frac{\exp^{x_i^t \prime \beta^k}}{1 + \exp^{x_i^t \prime \beta^k}} \end{aligned}$$

After simple algebraic manipulations, we find that $\gamma(\beta^k, x_i^t) = log(\frac{\pi(x_i^t)}{1-\pi(x_i^t)})$. We can obtain the optimal $\beta^k$ explicitly by solving equation 5.5 iteratively.

$$\begin{aligned} \beta^{k^{m+1}} &= \beta^{k^m} + (\sum_t w_{kt} x^t \prime \mathcal{Q}^{t^m} x^t)^{-1} (\sum_t w_{kt} x^t \prime \mathcal{Q}^{t^m} \Delta^t (y^t - \mu^t)) \quad (5.5) \\ \mathcal{Q}^{t^m} &= \{_d \mu_i^{t^m} (1 - \mu_i^{t^m})\} \\ \Delta^{t^m} &= \{_d \frac{1}{\mu_i^{t^m} (1 - \mu_i^t)^m}\} \\ \mu_i^{t^m} &= \pi(x_i^t) = \frac{\exp x_i^t \prime \beta^{k^m}}{1 + \exp x_i^t \prime \beta^{k^m}} \end{aligned}$$

Similar to regularization approaches, the multi-task generalized linear model also gives us biased estimations. Unlike other regularization methods to penalize all task at once, our method helps us select the related tasks and pool the parameter space for related tasks together without penalizing too much on unrelated ones. Variable selection can be very important for generalization especially in high dimensional problems. Unlike other regularization methods, we hope to achieve the variable selection step by discovering the statistical insignificant estimators through bootstrapping methods.

## 5.3 Simulations

### 5.3.1 Prediction Performance

In order to demonstrate predication performances, we compare testing errors, sum of square differences, among multi-task, individual and pooled learning method on many synthetic data sets, since all three approaches share the same model structures. Penalized approach is not included in this comparison, because there exists no simple procedures for the existing multi-task penalized approaches to uncover the unrelated tasks within the given information. We keep only limited number of tasks for this study to have a clear picture of relationship among all tasks. In the first study, we have total five tasks. When the responses are numeric variables with given weight information, regression approaches are employed in the simulation. We simulated the following situations with different signal to noise ratio by adjusting error variance. For each task, we have $x_i^k = (x_{i1}^k, \cdots, x_{i8}^k) \sim N(\overrightarrow{0}, I_8)$. The first three tasks has parameter as $\beta^1 = \beta^2 = \beta^3 = (1,1,1,1,0,0,0,0)$, while the last two tasks have the same parameter as $\beta^4 = \beta^5 = (0,0,0,0,1,1,1,1)$ with block weight matrix. Predictor $y_i^k$ is generated by $x'\beta +$ an error term with variance $\epsilon = (0.5, 1, 2.5, 5)$ for different signal to noise ratios. During iterations, each task is randomly divided into a testing (around 30 percent)and a training set with total 60 observations. Performance is evaluated based on the sum of square distances of all testing sets.

Figure.5.1 boxplot is based on 200 cross validations of testing errors from all three methods under different signal to noise ratios. It is easy to see, pooled regression (blue) always gives the worst performance for all error variances in the simulation since there exists unrelated tasks within the given information. When the signal to noise ratio is high, the least square method (green) have a comparable testing error

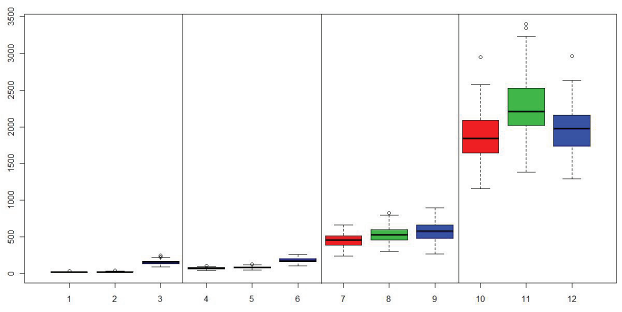


Figure 5.1: Box plot of testing errors over 200 cross validations. Red, green and blue box plots are constructed based on testing errors from multi-task, individual training and pooled methods. From the left column to the right column, signal to noise ratio decreases.

Table 5.1: Ratios of medians testing errors

| Methods | Multi-task | Independent training | Pooled training |
|---|---|---|---|
| $var(\epsilon) = .5$ | 1 | 0.921854 | 0.116384 |
| $var(\epsilon) = 1$ | 1 | 0.856275 | 0.384252 |
| $var(\epsilon) = 2.5$ | 1 | 0.850344 | 0.659147 |
| $var(\epsilon) = 5$ | 1 | 0.861968 | 0.960116 |

as multi-task learning method (red). As signal to noise ratio decreases, from the left column to the right column, multi-task learning method have lower testing errors than individual method by borrowing strength from multiple tasks. The outstanding prediction rate can also be found in table 5.3.1, where we compute the radios of median testing errors. Each cell in the table is computed as the median of testing errors from the multi-task method over the median of testing errors from the other methods.

For each of the given learning methods, we also use bootstrapping confidence intervals to check if one can correctly uncover the sparsity structure of the parameter. In this study, we use the same data set with a weight matrix, constructed based on Gaussian Kernel. Figure 5.2 shows some bootstrapping confidence intervals for some estimated parameters for both multi-task and individual methods with variance of error equaled to one. Table 5.3.1 records the percentage of bootstrapping confidence

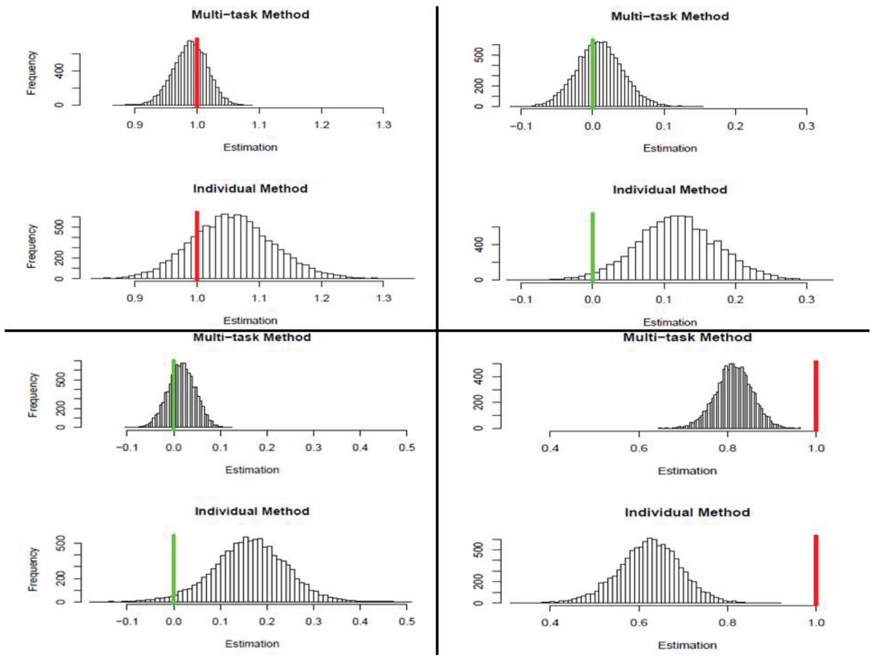


Figure 5.2: Bootstrapping results from both multi-task and individual methods for some of estimated parameters.

Table 5.2: Percentage of confidence interval correctly cover zero and average length of confidence intervals.

| Methods | Multi-task (Length) | Individual (Length) | Pooled (Length) |
|---|---|---|---|
| $var(\epsilon) = 0.5$ | 0.6 (0.132) | 0.65 (0.237) | 0 (0.162) |
| $var(\epsilon) = 1$ | 0.7 (0.179) | 0.6 (0.306) | 0.05 (0.217) |
| $var(\epsilon) = 2.5$ | 0.5 (0.391) | 0.55 (0.717) | 0.15 (0.299) |
| $var(\epsilon) = 5$ | 0.5 (0.875) | 0.55 (1.631) | 0.35 (0.612) |

interval which correctly cover the right sparse structure and the average length of all confidence intervals for estimations. Based on the table, the multi-task learning method is able to uncover the corrected parameter information with reasonable interval lengths.

Besides numerical responses, we also apply the multi-task learning method to 0/1 responses, $y_i^k$ follows Binomial $(x_i^{k\prime}\beta^k)$. In this case, we compare the multi-task model to individual and pooled logistic models under different generalized signal to noise ratios. In Gabriela's work (*Czanner et al.* (2008)), the generalized signal to noise

Table 5.3: Ratios of medians of sum of testing errors

| Methods | Multi-task | Independent training | Pooled training |
|---|---|---|---|
| $var(\epsilon) = .5$ | 1 | 0.921854 | 0.116384 |
| $var(\epsilon) = 1$ | 1 | 0.856275 | 0.384252 |
| $var(\epsilon) = 2.5$ | 1 | 0.850344 | 0.659147 |
| $var(\epsilon) = 5$ | 1 | 0.861968 | 0.960116 |

ratios are defined as

$$\frac{Dev(y, f(x, \hat{\beta}_0)) - Dev(y, f(x, \hat{\beta}))}{Dev(y, f(x, \hat{\beta}))}$$

where deviance is a measure of the distance between the fitted model and observed data y. $\beta_0$ is the component of the mean unrelated to the signal while $\beta$ is the component related to the signal. Figure 5.3 shows pooled logistic regression model perform the best when the generalized signal to noise ratios is small. As the signal increases, the multi-task learning model starts to have a better performance. Table 5.3.1 also contains ratio of median testing error among all methods.

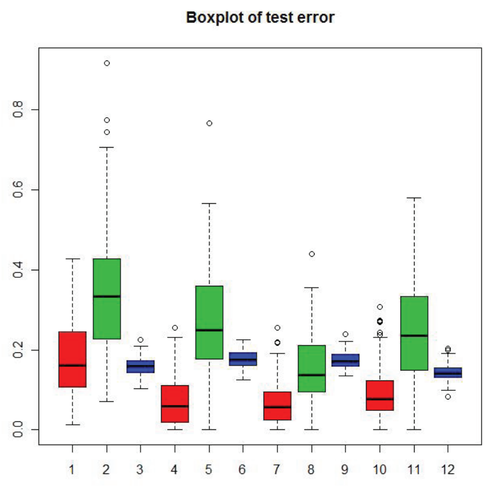


Figure 5.3: Box plot of testing errors over 200 cross validations under different generalized signal to noise ratios. Red, green and blue box plots are constructed based on testing errors from multi-task, individual training and pooled methods. The estimated generalized signal to noise ratios range from 5 to 50.

Table 5.4: average sum of square distances SSD

| Methods | Sum of square differences |
|---|---|
| Learning with weight 1 | 169.2043 |
| Learning with weight 2 | 35.66681 |
| Learning with weight 3 | 318.1389 |

Similar to other multi-task methods, our multi-task model give a better preference for both numerical and categorical responses based on a corrected information on relationship among tasks. However, we often do not have the most accurate information in real world applications. We demonstrate the importance in identifying the optimal weight matrix, $W$, based on simulations. Next, we have total four tasks where all $x_i^k = (x_{i_1}^k, \cdots, x_{i_8}^k) \sim N(\overrightarrow{0}, I_8)$ and responses are numerical variable $\epsilon_i^k \sim N(0, 0.5)$. Each task is divided into a testing and a training set with 30 observations. Each performance is once again evaluated based on the sum of square distances on testing sets. Three constant weight matrixes are applied to the case. The true parameter space can be correctly captured by the second weight matrix.

$$w_1 = \begin{bmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{2} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{2} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{2} \end{bmatrix} \quad w_2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad w_3 = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

We record the average sum of square distances over 200 iterations under different weight matrices in the table 5.3.1. Compare multi-task learning method's testing error across different weight matrices. When the weight matrix closely captures the structure of underlying parameter space, the learner will have the best performance. In real world applications, even though we often have some knowledge about relationships among tasks, we do not have accurate information to fully capture the true relationship among tasks. Besides comparing prediction performance on imprecise weight structures, we perturb the correct weight matrix by adding noise information

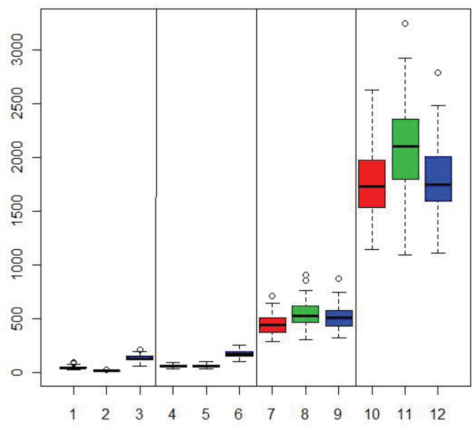


Figure 5.4: Box plot of testing errors over 200 cross validations from weight with white noise simulation. Red, green and blue box plots are constructed based on testing errors from multi-task, individual training and pooled methods.

to the observed weighted matrix. In the following example, we still have the same setting as in the first simulation where we have total five tasks. The parameter for the first three tasks is $\beta^1 = \beta^2 = \beta^3 = (1, 1, 1, 1, 0, 0, 0, 0)$. The last two tasks have the same parameter as $\beta^4$=$\beta^5$=$(0, 0, 0, 0, 1, 1, 1, 1)$. We add a white noise term, following uniform $(0, \frac{1}{3})$, to all elements of the correct constant weight matrix before standardization to create a new weight matrix. It is easy to show variance of estimated parameters for numerical responses can still be bounded by a function of variance of $\epsilon$ and variance of noise on the weight matrix. The estimated parameter follows a product distribution. In Figure 5.5, even with noise in the relationship among tasks, multi-task learning method still has outstanding performances. In table 5.3.1, we once again compare medians of testing errors among three methods. Since there exists additional noise within weight information, the individual method has a slightly better performance compare to other methods when signal to noise ratios are large.

Table 5.5: Ratios of medians of sum of testing errors

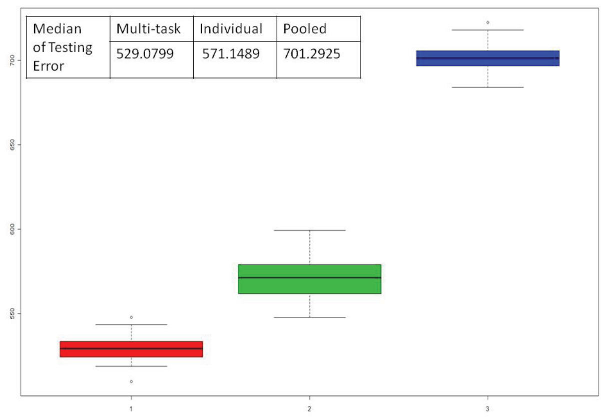
| Methods | Multi-task | Independent training | Pooled training |
|---|---|---|---|
| $var(\epsilon) = .5$ | 1 | 2.710279 | 0.367452 |
| $var(\epsilon) = 1$ | 1 | 1.037603 | 0.368603 |
| $var(\epsilon) = 2.5$ | 1 | 0.820177 | 0.866236 |
| $var(\epsilon) = 5$ | 1 | 0.851348 | 0.972207 |

## 5.4 Applications

We apply multi-task learning method to a movie rating data set to check performance under a real world situation. The goal of the study is to predict movie ratings from 514 users. In this application, each user is treated as one task, which contains 30 to 150 rated movies. Besides past movie ratings, we also have information on name, year, length, budget, rating from IMDB database and number of votes for each movie. For each user or task, we have information on some intrinsic attributes, such as user's age, gender, occupation and location. Since the rating ranges from 1 to 5 with a balanced distribution, we treat the movie rating as a numerical variable.

In this paper, we capture the relatedness among tasks by constructing the weight matrix based on common movie ratings. For example, if user $A$ and user $B$ both watched and rated "Toy Story", then the $(A, B)^{th}$ element in the weight matrix will be defined as $W_B^A$=$W_A^B$= $\mathcal{S}_\lambda(\texttt{Rate}^A_{\texttt{Toy story}}, \texttt{Rate}^B_{\texttt{Toy story}})$, where $\mathcal{S}$ can be any kernel function. Since we treat the movie rates as numerical variables for this study, we employed Gaussian kernel to construct the similarity between two tasks. Figure 5.5 shows a box plot of total average testing errors from 514 users for multi-task, individual and pooled methods over 50 cross validations. Due to heterogeneity of distributions, pooled method has the worst prediction performance. By effectively utilizing information obtained from users with similar tastes, multi-task has the best performance in predicting users movie ratings. On average, multi-task increases total prediction accuracy for 514 users by around 5 percent.

Apart from comparing overall predication error rate for all three methods, we also



| Median of Testing Error | Multi-task | Individual | Pooled |
|---|---|---|---|
| | 529.0799 | 571.1489 | 701.2925 |

Figure 5.5: Box plot of testing errors from 514 users over 50 cross validations.

take a closer look at performance for each user for both multi-task and individual methods. Figure 5.6 is a predication differences per user plot, where x-axis represents users ranked by number of rated movies and y-axis stands for predication differences between multi-task and individual methods. Negative value in the y-axis means that multi-task outperforms individual method for a particular user. It is easy to see that multi-task tends to have a better performance in predicting movie ratings for users with a few past ratings.

For statistical inference, we employ the bootstrapping method to construct confidence intervals for all users. Table 5.4 contains bootstrapping confidence intervals for three users with different group sizes, ( 34, 44, 113). The first column constitutes confidence bands for the user with the least observation and the last column constitutes information for user with the most observation. The second user and the third user rate 11 common movies. The first user and the third user only rate one movie in common. There are no common rated movies between user one and two. The bootstrapping confidence interval also reviews some degrees of similarity in parameters for user two and three.

Along with having forecasting models for predicting future movie ratings for 514 users, we also create a relationship network, which captures their movie preferences.

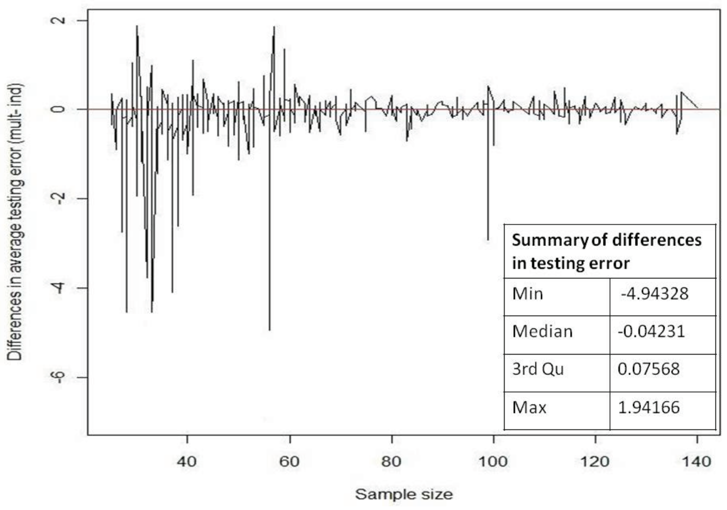


Figure 5.6: Closer look at testing errors from 514 users for multi-task and individual learning methods.

Table 5.6: Ratios of medians of sum of testing errors

| C.I. | User One | User Two | User Three |
|---|---|---|---|
| Year | ($2.34\times10^{-4}$, $3.68\times10^{-4}$) | ($4.69\times10^{-4}$, $6.04\times10^{-4}$) | ($9.52\times10^{-4}$, $1.02\times10^{-3}$) |
| Length | ($-1.01\times10^{-3}$, $4.48\times10^{-4}$) | ($2.07\times10^{-3}$, $3.66\times10^{-3}$) | ($6.71\times10^{-4}$, $1.78\times10^{-3}$) |
| Budget | ($-3.40\times10^{-9}$, $-1.27\times10^{-9}$) | ($-1.28\times10^{-9}$, $7.40\times10^{-10}$) | ($5.12\times10^{-9}$, $7.28\times10^{-9}$) |
| Rating | ($4.07\times10^{-1}$, $4.47\times10^{-1}$) | ($2.99\times10^{-1}$, $3.44\times10^{-1}$) | ($2.28\times10^{-1}$, $2.75\times10^{-1}$) |
| Votes | ($-7.27\times10^{-6}$, $-2.71\times10^{-6}$) | ($1.64\times10^{-6}$, $5.08\times10^{-6}$) | ($-1.02\times10^{-6}$, $2.42\times10^{-6}$) |

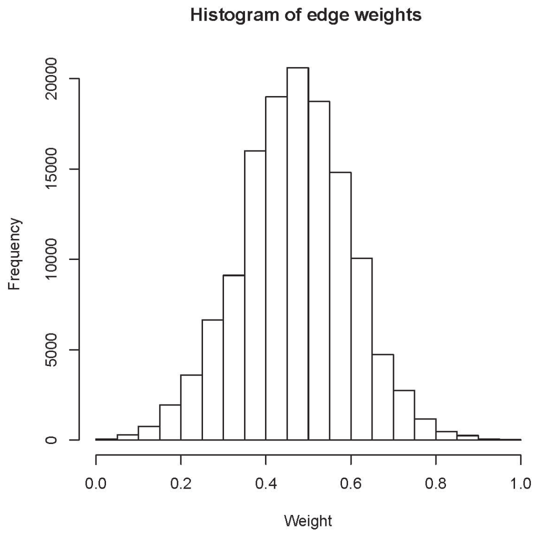


Figure 5.7: The edge weight distribution of movie preference network.

Figure 5.8 shows movie preference networks for all 514 users in the application, where nodes are users and edges are similar movie preferences. The weight of edge represents the similarity in movie preferences. Figure 5.7 is the histogram of edge weight. Node color represents the age group of a user. Yellow , orange and red stand for age group below age 25, 25-45 and above 45 groups. From top to bottom in Figure 5.8, the same movie preference network reviews less connections based on degrees of similarity. The second graph only contains over 800 connections with weight over 0.8. In this picture, we use a force directed layout based on edge weight. We can identify more similar movie preferences within the same age group in the same color connected nodes shown in the second network. There exists more similarities in movie tastes for 25-35 (orange) and above 45 (red) groups. The younger age group has the least similarity with other the two groups.

## 5.5 Conclusions and Future Work

In this paper, we propose a generalized linear model for multi-task learning problem by capturing the relatedness among tasks by a kernel weight matrix. We demonstrate the performance of the multi-task generalized linear model on multiple synthetic data sets. Even with noisy weight information, we also show superior prediction rates for multi-task learning model. Finally, we apply our multi-task learning model on a movie rating application. To explore its benefits, there are still many areas of multi-task learning problem waiting to be solved. The next step would be to apply our multi-task generalized linear model to different applications.

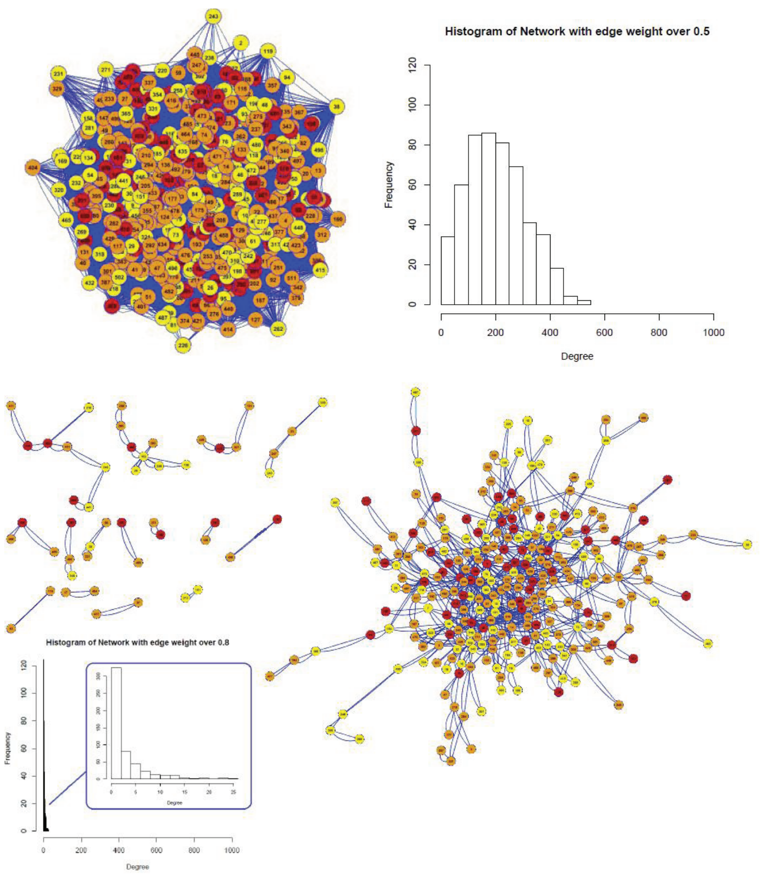


Figure 5.8: Movie Preference Network, where nodes are users and edges are similar movie preferences.From the top network to the bottom network, we select edges with high similarity level. Node color stands for different age groups (below age 25 = yellow, 25- 35 age group = orange , above 45 group = red).

CHAPTER VI

# Conclusion and Discussion of Future Work

This thesis addressees various aspects of visual analytics. Chapter 2 describes a visual algebraic framework for visual analytic process. While chapter 3 constructs an uncertainty model for querying based on this visual analytic algebra. Chapter 4 introduces a dimension reduction method based on penalized singular value decomposition to provide an effective overall visual representation for visual analytics. Chapter 5 proposes a statical learning method based on both observations and relationships among other related data to understand data on the individual level. All chapters consider different aspects of analyzing data set with network information. Next, we briefly discuss a number of future directions.

## 6.1 Network Data Analysis and Practical Tools

The data analytic process is not a straight line linked by a single procedure, it often involves much circling back, and looking at the workflow to assess where the user has been and where she should go next. Often, one needs to stop at the current findings to reinterpret them in terms of what we has been learned. Not only will our models of the data change, but also our visualizations and even the form of the data itself. *Velleman* (1997) captures this spirit well with his aphorism "The process of data analysis is one of parallel evolution. Interrelated aspects of the analysis evolve together, each affecting the others".

While presented separately, the dimension reduction and the multi-task learning method of this thesis work together to build our understanding of the network data from different aspects for one single analytic step. To balance our curiosity visually, trace our past discoveries consistently and guide our future analysis fully, we need the visual algebraic framework to encourage scepticism and remind us to question for future exploration. It is a tool to combine the human ability to see patterns in network data, and the innate desire to uncover stories about data.

It is difficult to judge the impact of all work. Based on the user experiment, the visual analytic algebra has high reproducibility to cope and repeat existing network graphs. Based on synthetic and real world applications, both dimension reduction and multi-task learning methods demonstrate their outstanding performances.

## 6.2 Future Work

My future work will still be evolving around the topic of visual analytics. I am currently developing a more effective tool for our visual algebra and extending the uncertainty model to incorporate the uncertain graph structure. In addition, I improve and implement the dimension reduction technique into a existing visualization system, such as R and Cytoscape. There are a number of functions that support user developed packages for both systems. Moreover, I am planing to extend the dimension reduction method to a data mining method for high dimensional data with network structures. By incorporating the additional network structure, we hope to improve the predication rate for all observations.

This thesis has presented many practical methods that support different aspects of visual analytics, improving our ability to explore data both statistically and visually. The methods are not just computational, but also provide a mental framework that supports a data visual analytic process and stresses a mixture of human and machine intelligence to progressively build our understanding of complex data.

# BIBLIOGRAPHY

# BIBLIOGRAPHY

Anderson, J. R. (1995), *Learning and Memory*, Wiley and Sons Inc.

Argyriou, A., T. Evgeniou, and M. Pontile (2006), Multi-task feature learning, *Journal of Machine Learning*, *10*, 243–272.

Argyriou, A., C. Micchelli, M. Pontile, and Y. Ying (2008), A spectral regularization framework for multi-task structure learning, *Neural Information Processing Systems*.

Auber, D., Y. Chiricota, F. Jourdan, and G. Melanc (2003), Multiscale visualization of small world networks, *In Proceedings of the IEEE Symposium on Information Visualization*, pp. 75–81.

Bakker, B., and T. Heskes (2003), Task clustering and gating for bayesian multi-task learning, *Journal of Machine Learning Research*, *4*, 83–99.

Batagelj, V., and A. Mrvar (2009), Pajek program for large network analysis, *Home page: http://vlado.fmf.uni-lj.si/pub/networks/pajek/*.

Baxter, J. (2000), A model of inductive bias learning, *Journal of Artificial Intelligence Research*, *12*, 149–198.

Ben-David, S., and R. Schuller (2003), Exploiting tasks relatedness for multiple task learning, *Proceedings of Computational Learning Theory (COLT)*, *37*, 373–384.

Borg, I., and P. Groenen (1997), *Modern Multidimensional Scaling: theory and applications*, Springer Series in Statistics.

Breiman, L., and J. Friedman (1997), Predicting multivariate responses in multiple linear regression, *Journal of the Royal Statistical Society,Series B*, *59(1)*, 3–54.

Caponnetto, A., C. Micchelli, M. Pontile, and Y. Ying (2008), Universal multi-task kernels, *Journal of Machine Learning Research*, *9*, 1615–1646.

Chen, J., J. Liu, and J. Ye (2010), Learning incoherent sparse and low-rank patterns from multiple tasks, *KDD*.

Choo, J., S. Bohn, and H. Park (2009), Two-stage framework for visualization of clustered high dimensional data, *Visual Analytics Science and Technology*, pp. 67–74.

Collaboration, T. C. (2006), *Cytoscape Users Manual*, Institute for Systems Biology and University of California San Diego.

Costa, L., F. Rodrigues, G. Travieso, and V. Boas (2007), Characterization of complex networks, *Advances in Physics*, *56*(1), 167–242.

Czanner, G., S. V. Sarma, U. Eden, and E. Brown (2008), A signal-to nosie ratio estimator for generalized linear model systems, *World Congress on Engineering*.

de Leeuw, J., and G. Michailidis (2000), Optimization transfer using surrogate objective functions, *Journal of Computational and Graphical Statistics*, *9*(1), 26–31.

Elmqvist, N., T. N. Do, H. Goodell, N. Henry, and J. Fekete (2008a), Zame interactive large-scale graph visualization, *In Proceedings of the IEEE Pacific Visualization Symposium 2008*, pp. 215–222.

Elmqvist, N., P. Dragicevic, and J.-D. Fekete (2008b), Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation, *IEEE Transactions on Visualization and Computer Graphics*, *14*(6), 1539–1148.

Evgeniou, T., and M. Pontil (2004), Regularized multi-task learning, *KDD' 04*.

Gene H. Golub, C. F. V. L. (1996), *Matrix computations*, third ed., Johns Hopkins studies in the mathematical sciences.

Grammel, L., M. Tory, and M.-A. Storey (2010), How information visualization novices construct visualizations, *IEEE Transactions on Visualization and Computer Graphics*, *16*(6), 943–952.

Grinstein, G., C. Plaisant, S. Laskowski, T. O'Connell, J. Scholtz, and M. Whiting (2008), Vast 2008 challenge: Introducing mini challenges, *Proceedings of IEEE Symposium*, *1*(1), 195–196.

Hanrahan, P. (2006), Vizql a language for query, analysis and visualization, *SIGMOD*.

Heer, J., S. K. Card, and J. A. Landay (2005), Prefuse: a toolkit for interactive information visualization, *In CHI 2005 Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 421–430.

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

Heskes, T. (2000), Empirical bayes for learning to learn, *Proceedings of ICML-2000,ed Langley,P.*, pp. 367–374.

Holten, D. (2006), Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data, *IEEE Transactions on Visualization and Computer Graphics*, *12*(5), 741–748.

Hornik, K., and A. Gebhardt (1998), *MASS Package*, R User Manual, CRAN.

Ideker, T., V. Thorsson, J. A. Ranish, et al. (2001), Integrated genomic and proteomic analyses of a systematically perturbed metabolic network, *Science*, *292*(4), 929–934.

I.Kecskes, and T. Papp (2000), *Foreign language and mother tongue*, Hillsdale.

J.D., B. A. Brown, and R. Cocking (1999), *How people learn: brain, mind, experience and school*, National Academy Press.

Jolliffe, I. (2002), *Principal Component Analysis*, second ed., Springer Series in Statistics.

Laramee, R., and R. Kosara (2007), Challenges and unsolved problem, *In Human-Centered Visualization Environments*, pp. 231–254.

Lounici, K., M. Pontile, A. Tsybakov, and S. vande Geer (2009), Taking advantage of sparsity in multi-task learning, *Conference on Learning Theory.*

Ma, K.-L. (2003), Visualizing time-varying volume data, *IEEE Computing in Science and Engineering*, *5*(3), 34–42.

McLachlan, G. J. (2004), *Discriminant analysis and statistical pattern recognition*, Wiley series in probability and mathematical statistics.

Michailidis, G. (2006), *Data Visualization Through Their Graph Representations.*

Miriah Meyer, T. M., and H. Pfister (2010), Multeesum: A tool for comparative spatial and temporal gene expression data, *IEEE Transactions on Visualization and Computer Graphics*, *1*(2), 99–108.

Moustafa, W. E., G. M. Namata, A. Deshpande, and L. Getoor (2011), Declarative analysis of noisy information networks, in *ICDE Workshop on Graph Data Management: Techniques and Applications.*

Muelder, C., and K.-L. Ma (2008), Rapid graph layout using space filling curves, *IEEE Transactions on Visualization and Computer Graphics*, *14*(6), 1301 – 1308.

Obozinski, G., B. Taskar, and M. Jordan (2006), Multi-task feature selection, *Technical Report.*

Obozinski, G., M. Wainwright, and M. Jordan (2008), Union support recovery in high-dimensional multivariate regression, *Technical Report.*

Obozinski, G., B. Taskar, and M. Jordan (2009), Joint covariant selection and joint subspace selection for multiple classification problem, *Statistics and Computing.*

Oesterling, P., G. Scheuermann, S. Teresniak, G. Heyer, S. Koch, T. Ertl, and G. H. Weber (2010), Two-sage framework for a topology-based projection and visualization of classified document collections, *IEEE Symposium on Visual Analytics Science and Technology*, pp. 91–98.

Olga Troyanskaya, M. C., et al. (2001), Missing value estimation methods for dna microarrays, *Biostatistics*, *17*(6), 520–525.

Olson, J., and G. Olson (1990), The growth of cognitive modeling in human computer interaction since goms, *Human computer interaction*, *5*, 221–265.

Pontile, M., T. Evgeniou, and A. Argyriou (2007), Convex multi-task feature learning, *Journal of Machine Learning*, *10*, 243–272.

Raudenbush, S., and A. Bryk (2002), *Hierarchical Linear Models*, Saga Publications, Inc.

Schneidewind, J., and H. Ziegler (2006), Challenges in visual data analysis, *Proc. Int'l Conf. Information Visualization (IV)*, pp. 9–16.

Shaverdian, A., H. Zhou, G. Michailidis, and H. Jagadish (2009a), Algebraic visual analysis: The catalano phone call data set case study, *VAKD 09*.

Shaverdian, A. A., H. Zhou, G. Michailidis, and H. Jagadish (2009b), Algebraic visual analysis: the catalano phone call data set case study, *Proc. ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery*.

Shojaie, A., and G. Michailidis (2009), Analysis of gene sets based on the underlying regulatory network, *Journal of Computational Biology*, *6*(3), 407–426.

Stasko, J., C. Grg, Z. Liu, and K. Singhal (2007), Jigsaw: Supporting investigative analysis through interactive visualization, *Proc. IEEE Symp. Visual Analytics Science and Technology*, pp. 131–138.

Thomas, J., and K. Cook (2005), Illuminating the path: The research and development agenda for visual analytics, *IEEE Computer Society*.

Tibshirani, R. (1996), Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*, 267–288.

Travers, J., and S. Milgram. (1969), An experimental study of the small world problem, *Sociometry*, *32*(4), 425–443.

Tulip, D. A. (2004), A huge graph visualization framework, *In Graph Drawing Software, Mathematics and Visualization*, pp. 105–126.

VAST (2009), Vast 09 challenge dataset, *http://hcil.cs.umd.edu/localphp/hcil/vast/index.php*.

Velleman, P. F. (1997), The philosophical past and the digital future of data analysis 375 years of philosophical guidance for software design on the occassion of john w. tukeys 80th birthday.

Viau, C., M. J. McGuffin, Y. Chiricota, and I. Jurisica (2010), The flowvizmenu and parallel scatterplot matrix: Hybrid multidimensional visualizations for network exploration, *IEEE Transactions on Visualization and Computer Graphics*, *16*(6), 1100–1108.

Weaver, C. (2008), Multidimensional visual analysis using cross-filtered views, pp. 163–170.

WIKIPEDIA (2011), Scoring algorithm.

Witten, D. M., R. Tibshirani, and T. Hastie (2009), A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis, *Biostatistics*, *10*(3), 515–534.

Wong, P., H. Foote, G. C. Jr., P. Mackey, and K. Perrine (2006), Graph signatures for visual analytics, *IEEE Transactions on Visualization and Computer Graphics*, *12*(6), 1399–1413.

Yuan, X., P. Guo, H. Xiao, H. Zhou, and H. Qu (2009), Scattering points in parallel coordinates, *IEEE Transactions on Visualization and Computer Graphics*, *15*(6), 1001–1008.

Zhang, J., Z. Ghahramani, and Y. Yang (2008), Flexible laten variable models for multi-task learning, *Journal of Machine Learning*.