Visual Analytics with Biclusters:
Exploring Coordinated Relationships in Context

Maoyuan Sun

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Applications

Christopher L. North, Chair
Naren Ramakrishnan
Kurt Luther
Danfeng (Daphne) Yao
Nathalie H. Riche

July 27, 2016
Blackburg, Virginia

Keywords: biclusters, coordinated relationship, edge bundling, visual analytics

Copyright 2016, Maoyuan Sun
Exploring coordinated relationships is an important task in data analytics. For example, an intelligence analyst may want to find three suspicious people who all visited the same four cities. However, existing techniques that display individual relationships, such as between lists of entities, require repetitious manual selection and significant mental aggregation in cluttered visualizations to find coordinated relationships.

This work presents a visual analytics approach that applies biclusters to support coordinated relationships exploration. Each computed bicluster aggregates individual relationships into coordinated sets. Thus, coordinated relationships can be formalized as biclusters. However, how to incorporate biclusters into a visual analytics tool to support sensemaking tasks is challenging. To address this, this work features three key contributions: 1) a five-level design framework for bicluster visualizations, 2) BiSet, highlighting bicluster-based edge bundling, seriation-based multiple lists ordering, and interactions for dynamic information foraging and management, and 3) an evaluation of BiSet.

This work was supported in part by a grant from L-3 Communications, and NSF grants CCF-0937133, IIS-1218346 and IIS-1447416.
Acknowledgments

First, I thank Chris North for offering me an inspiring and encouraging mentorship. For me, Chris is a great advisor, not only for research but also for my life. I have learnt so many from him far beyond listed here, such as how to think from multiple perspectives, write papers, present results, give talks, collaborate with others and even think about the meaning of life. His advice, assistance and patience leads me passing through many difficulties, which finally leads to this dissertation.

I also thank my dissertation committee members, Naren Ramakrishnan, Kurt Luther, Danfeng (Daphne) Yao, and Nathalie Riche, for advice and suggestion to improve this dissertation. Naren Ramakrishnan provided great support for me to understand the concept of bicluster and present it in an easy way. Kurt Luther helped me with the design of the system and gave me useful feedback about its initial version. Danfeng (Daphne) Yao offered good advice for me to present this dissertation work to more broad audience. Nathalie Riche led helpful discussion about the design of the evaluation study and asked inspiring questions to help me think deep of the study.
Many thanks also go to my InfoVis lab mates, and friends at the discovery analytics center, the center for human-computer interaction and the computer science department, whom I have been blessed to work with. Thanks to Peng Mi, Hao Wu, Lauren Bradel, Junyang Chen, Hao Zhang, Bin He, Junpeng Wang, Xin Chen, Jessica Self, Nathan Self, Ji Wang, Caleb Reach, and many others. Thanks for the help and encouragement.

I greatly thank those who funded this research, including NSF and L-3 Communications. Also thanks to Gregorio Convertino, Mark Detweiler, Liam Friedland, and other UX team members at Informatica for supporting my interesting explorations along the way.

Sincere appreciation and thanks give to my wife, Yujun Liu, my family and family-in-law. Thanks Yujun for always being there with me, understanding me, and the sacrifice for the family. Thanks Mom and Dad who always taught me to be the best of myself, independent and responsible.

Most of all, I thank God for all his grace with me, leading me to a wonderful advisor, inspiring me with ideas, providing me with great family, friends and collaborators, and bringing peace to my heart for everything. This dissertation results from all great gifts from God.
## Contents

**Chapter 1**

**Introduction**

1.1 Research Overview ......................................................... 4

1.2 Organization ................................................................. 5

**Chapter 2**

**Related Work**

2.1 Biclustering ................................................................. 7

2.2 Lattice ................................................................. 10

2.3 Similarity between Biclusters ........................................... 11

2.4 Chaining Biclusters ....................................................... 12

2.5 Visual Links and Edge Bundling ....................................... 14

2.6 Seriation ................................................................. 15

2.7 Max Entropy Principle based Bicluster Evaluation ............... 17
Chapter 3 Five-level Design Framework

3.1 Five Levels of Relationships .................................................. 19

3.2 The Five-level Design Framework ........................................... 23

  3.2.1 Design Choices for Entity and Group Levels ....................... 24

  3.2.2 Design Choices for Bicluster Level .................................... 27

  3.2.3 Design Choices for Chain Level ....................................... 38

  3.2.4 Design Choices for Schema Level ..................................... 42

3.3 Four-Level of Interaction Design .......................................... 44

3.4 Summary ................................................................. 46

Chapter 4 BiSet

4.1 Design Requirement Analysis .............................................. 50

  4.1.1 A Design Trade-off ....................................................... 50

  4.1.2 Layout Candidate Selection ........................................... 53

  4.1.3 Requirements Description with Identified Tasks ................... 55

4.2 BiSet Technique ............................................................ 57

  4.2.1 Data Level: Bicluster Discovery ..................................... 57

  4.2.2 Visual Level: Bundles as Objects and “In-between” ............... 58
4.2.3 Interaction: Exploring and Organizing Information 66

Chapter 5 Advanced Interactions to Scale Up 71

5.1 Requirement Description 72

5.2 Four Ways of Interacting with Biclusters 75

5.2.1 Seriation in Lists 75

5.2.2 Aggregation 80

5.2.3 Prioritization 85

5.2.4 Attraction 89

5.3 Usage Scenario 92

Chapter 6 Evaluation 98

6.1 Participants and apparatus 100

6.2 Data 100

6.2.1 Dataset Preparation with a Pilot Study 101

6.2.2 Dataset Generation for the Primary Study 104

6.3 Methodology 106

6.3.1 Data Analysis Method 108
6.4 Procedure ................................................................. 108
6.5 Data Collection ........................................................... 109
6.6 Hypotheses ................................................................. 110
6.7 Results .......................................................... 111
  6.7.1 Variance ............................................................... 111
  6.7.2 Accuracy .............................................................. 115
  6.7.3 Interaction Effort and Time Cost ......................... 116
  6.7.4 Summary of Performance Results ......................... 121
  6.7.5 Four Key Trade-offs ............................................... 122
6.8 Discussion .............................................................. 129
  6.8.1 Considering the Role of Machine Learning .......... 130
  6.8.2 Implication for Tools of Visualizing Relations .......... 131
  6.8.3 Study Limitation .................................................. 132

Chapter 7
Conclusion ................................................................. 133

  7.1 Research Contributions .............................................. 134
  7.2 Future Opportunities ................................................ 136
    7.2.1 Algorithm Comparison ......................................... 136

viii
List of Figures

1.1 An example of a bicluster that reveals a coordinated relationship between four students and three classes. ......................................................... 4

2.1 An example of a lattice that conceptually shows the relations of all possible biclusters identified from a data matrix on the top right. .............. 11

2.2 An example of a bicluster-chain with two biclusters, connected together based on their shared students. ......................................................... 13

3.1 An example of the matrix-based visualization to illustrate two biclusters mined from a gene-expression matrix. .............................................. 30

3.2 An example of finding a bicluster in Jigsaw’s List View. Yellow indicates entities that a user selected and orange indicates relevant entities corresponding to the selected one(s) based on word co-occurrence. .................. 33

3.3 An example of rearranging axes by switching axes BB and CC. .............. 36
3.4 An example of bicluster chain with three biclusters in Bixplorer.

3.5 A summary of the five-level design framework for bicluster visualizations.

4.1 An example of a bicluster, indicating a coordinated relationship between three people and four locations. (A) presents all connections between each pair of related entities from the two domains. (B) shows the result of bundling edges in this bicluster. (C) demonstrates the result of both bundling edges and grouping entities in this bicluster.

4.2 A detailed example to illustrate the Euler diagram problem that arises when visualizing the membership of entities in the domain of people shared by three biclusters. This problem indicates a key design trade-off: entity-centric versus relationship-centric.

4.3 A 2D space is sliced into two types of subspaces in a list view.

4.4 Three modes in the “in-between” layer for displaying edges. (A) is the edge only mode that shows all edges between related entities. (B) is the hybrid mode, which presents bundles with individual edges. (C) is the bundle only mode that just displays bundles.

4.5 A semantic edge bundling example in BiSet. (A) shows the original 16 edges. (B) After semantic edge bundling, there are 9 bundles.
4.6 Visual encodings in BiSet. (1), (2) and (3) present the normal state of an entity, a bundle and edges, respectively. (4) and (4’) show the selected state of an entity and a bundle with accumulated highlighting. (5) and (5’) present the mouseover state of an entity and a bundle. (6) shows accumulated highlighting of an entity. (7) presents the highlighting state of edges. (8) shows the accumulated highlighting of edges.

4.7 Dragging a bundle in the “in-between” layer. Entities associated with this bundle automatically move to their new positions.

4.8 The document view from bundles in BiSet. (A) shows bicluster ID, related document and associated entities. (B) shows the content of a document. (C) lists all document ID(s) in the data with a search function.

5.1 An example of organizing one bicluster-list and two entity-lists with two approaches. (A) gives the result of the greedy approach. (B) shows the result of the seriation approach.

5.2 An example of detailed visual encodings. (A), (B) and (C) present a bicluster, a merged bicluster, a merged bicluster (under selection), respectively. (D) and (E) show edges connecting merged biclusters with related entities, and those in (E) are highlighted. (F) and (G) show entities in the normal state and those in connection based highlighting state.
5.3 An example of automatic aggregation. (A) shows the layout after seriation. (B) is the result of applying aggregation to the layout in (A).

5.4 An example of visual encodings used in prioritization. (A) is a bicluster that users request for the two levels of evaluations. (B) and (B’) respectively shows the most surprising biclusters in the same list and in a neighboring list. (C) and (C’) presents the least surprising biclusters in the same list and in a neighboring list. (D) shows normal biclusters that are not highlighted. (E) and (F), respectively, show surprisingness based and connection based entity highlighting. (G) presents the links in the most surprising chain.

5.5 An example of attraction with manual aggregation. (A), (B) and (C) sequentially shows the visual attraction as users drag a bicluster. (D) and (E) shows manual bicluster aggregation with a similarity radar.
5.6 Key steps in finding two coordinated threat plots. (1): Using seriation to organize biclusters and entities and merging biclusters. (2): Splitting a merged bicluster, 4 normal biclusters appear and two of them are linked with A. Ramazi. (3) and (4): Evaluating related biclusters of the two biclusters noticed in (2) and their top ranked ones are the same. (5): From this shared top ranked bicluster, the most surprising chain leads to one threat plot. (6): From another bicluster in the most surprising chain identified in (5), its related biclusters in a neighboring bicluster-list are ranked. (7): The top ranked bicluster in (6) leads to another connected bicluster. (8): Similar biclusters of the bicluster noticed in (7) are shown in a similarity radar and one of them is connected with A. Ramazi. (9): From a merged bicluster, the most surprising chain leads to another threat plot.

6.1 Examples of three levels of overlaps based on shared entities.

6.2 A summary of total number of found biclusters, corresponding to each of the four types, in all experiment conditions.

6.3 Total number of entities visited, on average, per justified answer.

6.4 A summary of total counts of entity visits per justified answer on average.

6.5 A summary of average time (seconds) per justified answer for the four tasks.
6.6 An example of unreasonably justified answers in NS and BS. The selected entities are those inside the blue dotted boxes.  

6.7 An example of findings with different perception emphasis in their explanations. (A) shows an answer in NR, explained with emphasis on entity connectedness. (B) presents a finding in BR, with a coordinatedness oriented explanation.  

7.1 The existing paradigm of using biclusters for sensemaking.  

8.1 The dataset used for the NR condition in the primary study.  

8.2 The dataset used for the BR condition in the primary study.  

8.3 The dataset used for the NS condition in the primary study.  

8.4 The dataset used for the BS condition in the primary study.
# List of Tables

3.1 A Brief Summary of the Five-Level Design Framework. \ldots 22

4.1 A Summary of Identified Tasks in the Four Types of Exploration. \ldots 56

6.1 A summary of a dataset used in the pilot study. \ldots 102

6.2 A summary of the pilot study results. \ldots 103

6.3 A summary of the factor combination for each user task. \ldots 107

6.4 Average number of shared entities in submitted merged biclusters. \ldots 114

6.5 A summary of justified answers in all four tasks \ldots 116
Chapter 1

Introduction

Identifying coordinated relationships is an important task for data analytics in various fields. For example, intelligence analysts often examine large unstructured textual datasets to identify coordinated relations between different entity types (e.g., people, location, date, etc.) that might be evidence for collusion [68]. Bioinformaticians explore coordinated relations from expression and interaction datasets to identify groups of genes and/or proteins that are commonly expressed or regulated conditions and species [2, 103]. Analysts in cyber security trace coordinated relations between processes, hosts and network domains to detect distributed coordinated attacks [146]. Coordinated relationships are grouped relations between sets of entities of different types. Therefore, compared with simple relationships, coordinated relationships need more cognitive effort for exploration because of their complexity.

With training, analysts can manually identify and explore coordinated relations in data, but
with significant cognitive effort. This process usually involves three essential repetitive tasks: 1) identifying and extracting meaningful entities, 2) investigating entities to verify whether a set of entities are related to the same specific entity or entities, and 3) clustering or grouping entities based on their shared relationships. For example, in text analytics, to find three people who all visited the same four cities, analysts may read numerous documents, identify names and cities from the documents, compare many co-occurring people-city pairs among different scenarios, and test many possible combinatorial groupings of the pairs, to finally identify the four people who are all paired with the same five cities. Due to such repetitious tasks, analysts usually seek the help of visual analytics [30] to support their investigation.

Current visual analysis tools emphasize individual relationships and just display simple relations. This makes it difficult for analysts to easily see more complex relationships (e.g., coordinated relationship). Since existing techniques display individual relationships, such as between lists of entities, they require repetitious manual selection and significant mental aggregation in cluttered visualizations to find coordinated relationships.

For example, Jigsaw [117] provides a List View to support exploration of relationships between lists of entities (e.g., people, locations, dates, organizations, etc.). In the List View, Jigsaw applies visual links between related entities to show their connections and controls the shading of colors for entities to indicate their co-occurrence in documents. With these visual encodings, in Jigsaw, users can recognize relations between entities without much cognitive effort, but these relations are limited to simple individual ones (e.g., one person visited four cities). Users have to repetitiously click entities, visually check and mentally compare their
linked entities to identify coordinated relationships (e.g., three people visited four cities). Since Jigsaw’s List View does not provide clear clues on coordinated relations, users may have to manually test all possible entities before they finally find the meaningful ones. The problem gets even more complicated when composing or chaining multiple such coordinated relationships (e.g., did these people travel on the same dates?). This potentially forces users to solve a combinatorial problem of selection without much support. Thus, because of deficient clues to direct user selections, tools similar to Jigsaw have limited capabilities to support the exploration of coordinated relationship.

Visual analytics techniques can potentially better support this for sensemaking [101] by computationally finding complex relationships and revealing them in context. This enables analysts to see the complex relations with other data (e.g., entities in lists). Computation can ease combinatoric exploration through the use of effective data mining algorithms. Specifically, biclustering algorithms provide an efficient solution to identify coordinated relations.

Biclustering is a data mining technique that has been extensively used in bioinformatics, especially for gene expression data analysis [7, 25, 88–90, 102, 108, 126]. Biclusters, the computational outcome of biclustering algorithms, potentially provide a rich high-level abstraction that represents coordinated relationships between groups of entities of different types (e.g., a group of genes behave similarly under a group of conditions). In general, a bicluster can be considered a complete bipartite graph where every vertex of one set is connected to all vertices of another set. Specifically, a bicluster in a relation can be viewed as a bundling of individual relationships into a pair of sets. For instance, Figure 1.1 shows an
example of a bicluster which indicates four students taking the same three classes.

Figure 1.1: An example of a bicluster that reveals a coordinated relationship between four students and three classes.

1.1 Research Overview

While biclusters provide a good mathematical foundation to identify coordinated relationships, to support sensemaking, biclusters have to be made usable through interactive visualizations. However, how to design efficient, human perceptible and usable visual representations of biclusters with necessary interactions to assist sensemaking is still challenging.

To address this, three key research questions are listed as follows:

RQ1: What is the design space of bicluster visualizations?

(a) How can a design framework be built to inform the design of bicluster visualizations?

(b) What is the key design trade-off of bicluster visualizations?
RQ2: How can we instantiate the identified design options?

(a) How can biclusters be visualized based on the proposed design framework?

(b) What interactions can be applied to the visualized biclusters to support user exploration of coordinated relationships?

RQ3: How does the proposed bicluster visualization impact users’ exploration of coordinated relationships?

(a) Do bicluster oriented edge bundling and entity ordering, compared with traditional list representations, help users more efficiently explore coordinated relations?

(b) Comparing a traditional list view with BiSet, what are the trade-offs when using them for coordinated relationships exploration?

1.2 Organization

To address the above three questions, this document is organized as follows. Chapter 2 provides information of the key concepts used in this document and related research. These important concepts include: bicluster, lattice and bicluster-chains. Related research contains: bicluster similarity measurement, bicluster evaluation, edge bundling and seriation.

Chapter 3 addresses RQ1 with the result of a five-level design framework for bicluster visualizations, appearing in the publication [125]. This framework considers the relationship-space
involved in the exploration of coordinated relationships. The key design trade-off of bi-cluster visualizations, entity-centric versus relationship-centric, is identified and discussed in Section 4.1.1 of Chapter 4. In addition, Chapter 4 presents BiSet, based on previous publication [123], which addresses RQ2. Chapter 5 addresses advanced interactions to scale up BiSet, partially informed by the publication [137]. RQ3 is addressed through the results of a user study, which is presented in Chapter 6. Finally, Chapter 7 summarizes the key contributions of this work and future research opportunities. These summarized research opportunities are informed by the publication [124].
Chapter 2

Related Work

2.1 Biclustering

Clustering is a well-established concept, which has been comprehensively explored over the past fifty years [70]. The basic idea of clustering is that we are given $n$ points or entities in a given $m$-dimensional space and a distance or similarity function defined over that space. The goal is to identify subsets (clusters) of entities such that points within a cluster are more similar (or nearer) to each other than to points from other clusters.

Compared with the concept of clustering, biclustering is a relatively younger concept. The idea of biclustering (although not under this name) has existed since 1972 [57]. Biclustering generalizes the idea of clustering by simultaneously finding both subsets of entities and subsets of dimensions such that the selected entities are homogeneous (only) within the
selected dimensions. Biclustering thus treats the notion of points and dimensions more uniformly, which is different from clustering. Also, while clusters form a partition of the dataset (i.e., they are mutually exclusive and collectively exhaustive), biclusters can overlap and may not collectively span the entire matrix of relationships. If these two conditions are imposed, biclustering is also referred as co-clustering [36].

Starting with relations between entity sets, the notion of biclusters used in this paper is formalized as follows:

**Relations between two entity sets.** An entity set is a set of objects from a specific domain (e.g., person, location or date). Assume that entities have been extracted from datasets (e.g., documents) by using entity recognizers such as LingPipe [19] or similar tools. Given two entity sets $E$ and $F$, a (binary) relationship $R(E,F)$ between $E$ and $F$ is a subset of $E \times F$ (the Cartesian product of $E$ and $F$). In this case, $E$ is connected to $F$. It is useful to view $R$ as both a matrix and as a bipartite graph. In text analytics, $R$ can be used variously to model document co-occurrence, associations, or specific relations extracted by natural language processing. For instance, person $X$ can be related to organization $Y$ if they are mentioned in the same sentence, or if a dependency parse followed by a semantic labeling infers a “works-for” relationship between $X$ and $Y$.

**Bicluster.** A bicluster $(E', F')$ on $R(E,F)$ is defined as a set $E' \subseteq E$ and a set $F' \subseteq F$ such that $E' \times F' \subseteq R$. That is, there is a relationship between every element of $E'$ with every element of $F'$. A bicluster $(E', F')$ is thin if there is only one entity in either $E'$ or $F'$. 
Closed bicluster. A bicluster \((E', F')\) is closed if:

(i) For every entity \(e \in E - E'\), there is some entity \(f \in F'\) such that \((e, f) \notin R\), and

(ii) For every entity \(f \in F - F'\), there is some entity \(e \in E'\) such that \((e, f) \notin R\).

That is, adding an entity in \(E - E'\) or \(F - F'\) to the bicluster will violate the condition that defines a bicluster mentioned above. In other words, a closed bicluster is the bicluster to which no additional rows or columns can be added if it is represented in the form of matrix. Hence, a closed bicluster can be regarded as maximal in height and width (although the term “maximal bicluster” is sometimes reserved for other interpretations in the data mining community). In this paper, the notation of biclusters refers to closed biclusters.

Algorithms for bicluster mining typically aim to find closed biclusters. These algorithms (e.g., CHARM [141] and LCM [129]) function level-wise with regard to one domain (e.g., \(E\)), wherein they attempt to mine closed biclusters involving one entity of \(E\), then closed biclusters involving two entities of \(E\), and so on. The key parameter influencing such mining is the size of a bicluster in terms of the other domain (e.g., \(F\), also referred to as the minimum support threshold. The setting of this parameter is done heuristically by users; a low threshold will yield a plethora of biclusters whereas a stringent (high) threshold will yield few (or no) biclusters. Typically, users begin with a high threshold and gradually lower it until it yields a sufficient number of biclusters [141].
2.2 Lattice

Lattice [34] conceptually reveals relations among all possible biclusters identified from a data matrix. We begin the introduction of lattice with the concept of ordered sets. A partial order on a set $P$ is a binary relation $\leq$, such that for all $x, y, z \in P$, the relation meets: 1) reflexive ($x \leq x$), 2) antisymmetric (if $x \leq y$ and $y \leq x$, then $x = y$), and 3) transitive (if $x \leq y$ and $y \leq z$, then $x \leq z$). $P$ with the relation $\leq$ is called an ordered set, denoted as a pair $(P, \leq)$. Let $S \subseteq P$, an element $u$ is an upper bound of $S$, if $s \leq u$ for all $s \in S$, and an element $l$ is a lower bound of $S$, if $l \leq s$ for all $s \in S$. The least upper bound of $S$ is called the join of $S$, denoted as $\vee S$. The greatest lower bound of $S$ is called the meet of $S$, denoted as $\wedge S$. If $S = \{x, y\}$, we write $x \vee y$ for the join and $x \wedge y$ for the meet.

An ordered set $(P, \leq)$ is a lattice, if any two elements $x, y \in P$, the join $x \vee y$ and the meet $x \wedge y$ always exist. $P$ is a complete lattice if $\vee S$ and $\wedge S$ exist for all $S \subseteq P$. Any finite lattice is complete. $P$ is called a join semilattice, if only the join exists, and $P$ is called a meet semilattice, if only the meet exists. Let $\mathcal{P}$ denote the power set of $S$ (the set of all subsets of $S$), the ordered set $(\mathcal{P}(S), \subseteq)$ is a complete lattice, where the meet is given by set intersection, and the join is given by set union. A set of all possible biclusters mined from a data matrix is a meet semilattice [142]. Figure 2.1 shows an example of a lattice that reveals the relations among all biclusters identified from a data matrix. From this lattice (from bottom to top), we can find that the number of entities in one set decreases (see the set with numbers), as more entities are added to another set (see the set with characters).
Figure 2.1: An example of a lattice that conceptually shows the relations of all possible biclusters identified from a data matrix on the top right.

### 2.3 Similarity between Biclusters

*Jaccard index* (also known as *Jaccard similarity coefficient*) is a useful measure to evaluate similarity between two sets, and it has been used to determine the similarity between two biclusters [63, 77]. The Jaccard index for two given biclusters, $a$ and $b$, can be computed as follows:

\[
J(a, b) = \frac{|A \cap B|}{|A \cup B|} \quad (2.1)
\]

In equation (2.1), $A$ and $B$ respectively represent the set of all entities in bicluster $a$ and that
in bicluster \( b \). A bicluster can also be considered as a combination of two sets of entities from different domains (e.g., student and class). In this case, based on which set(s) we emphasize on (e.g., students only, class only, or both student and class), there are three possible types of similarity between two biclusters. For enabling flexible user adjustment on which set to emphasize and how much to emphasize, in this paper, we use the following weighted Jaccard index to measure the similarity between two given biclusters, \( a \) and \( b \).

\[
J_w(a, b) = w \cdot J_c(a, b) + (1 - w) \cdot J_r(a, b), \quad (0 \leq w \leq 1)
\] (2.2)

In equation (2.2), \( J_c(a, b) \) indicates the similarity between the two biclusters on one entity set (e.g., class), and \( J_r(a, b) \) reveals the similarity between them on another entity set (e.g., student). \( w \) is a user selected weight that represents how much one entity set is emphasized.

### 2.4 Chaining Biclusters

Since every bicluster is discovered in a single relation, it is possible to compose separately identified biclusters across two relations by (approximately) matching biclusters with the shared domains. Jin et al. presented this approach to identify compositional patterns in multi-relational datasets [73]. Figure 2.2 shows an example of a chain with two biclusters. One shows a coordinated relationship between four locations and three students, and the other presents relations between four students and three classes. They share three students. One possible insight that we may infer from this chain is: the three students may
Figure 2.2: An example of a bicluster-chain with two biclusters, connected together based on their shared students.

be labmutes, because they took the same set of classes and also visited the same four cities together (possibly for attending research conferences). By chaining biclusters across multiple relations, relationships from a diversity of domains can be bundled in a coherent manner. Results of such compositions can be read sequentially from one end to the other, which is similar to a story.

Chaining biclusters can be achieved by using similarity search algorithms and data structures, (e.g., the cover tree, an efficient data structure for calculating nearest neighbors [12]). For each unique domain (e.g., people, locations, dates, etc.), one cover tree can be defined. For every bicluster discovered, the set of rows and the set of columns within the bicluster are indexed into two corresponding cover trees. After all biclusters are indexed, similarity searches can be readily conducted to find closest overlaps to all identified biclusters [138], which works as the basis for chaining biclusters.
2.5 Visual Links and Edge Bundling

Visual links (e.g., edges in graphs) are important for assisting visual navigation [62] and indicate certain types of relations (e.g., causality [144]). By following links, users can navigate their foci from one part of a visualization to another, or across different visualizations [130] or applications [132]. This helps to direct users to potentially related content for comparison and evaluation [28, 119] or assist users to explore visually hidden (or being covered) content [49]. However, cases are not always optimistic. If too many edges exist, a visual layout (e.g., graph) will become a hairball of visual clutter [91]. Edge bundling is a useful technique to reduce visual clutter and reveal high level edge patterns by visually aggregating edges based on certain rules (e.g., force-directed model [66], image-based rule [128], geometry-based rule [32], etc.). However, there are two major problems with these edge bundling techniques: 1) spatial-based bundling in the visual level (losing relations in the data level), and 2) lack of interactions on edge bundles.

Traditional edge bundling techniques simply group edges based on spatial proximity (e.g., the position of nodes or edges), which may ignore some implicit relations in a dataset. Since visual adjacency is determined by layout algorithms (e.g., force-directed layout), rather than knowledge discovery algorithms (e.g., biclustering), bundling visually adjacent edges does not guarantee that a bundle of these individual relations reflect meaningful semantic insights from the dataset. To deal with such problems, a hierarchical edge bundling technique is proposed in [65] that bundles adjacent edges by considering the hierarchical relations in a dataset.
However, compared with coordinated relationships, hierarchical relationships are relatively simple because they can not reveal high level semantic insights implied by the coordination of individual relationships. Despite this, the hierarchical edge bundling technique inspires our design of BiSet that bundle edges based on coordinated relationship. This potentially enables users to infer semantic insights from these edge bundles.

Deficient interaction on edge bundles is another problem with existing edge bundling techniques. Such bundles have limited capabilities to support users exploring the space of relationship, although they help to reduce visual clutter. This partially results from the previous problem since edge bundles depend on the layout of nodes. If positions of nodes change, the existing bundles may also change. This means that these edge bundles are not stable. In BiSet, we map algorithmically discovered coordinated relationships to edge bundles. This assures that each bundle visually presents a certain coordinated relationship. Thus, bundles in BiSet are more independent from the layout of nodes than existing techniques. Based on this, BiSet allows interactions on edge bundles which enables users to manipulate bundles to forage related information, and organize spatializations for synthesis [5].

2.6 Seriation

Seriation is an exploratory combinatorial data analysis technique [87]. It permutes the order of objects to get a sequence where the regularity and pattern (e.g., clustering structure [135]) among the whole series can be well revealed. Seriation is commonly used to present patterns in
a matrix by permuting rows and columns (e.g., Bertifier [100], BiVoc [56] and Termite [27]).

Seriation in a matrix with $M$ rows and $N$ columns attempts to find orders of rows and columns that optimize some objective function. Finding all possible combinations of ordering rows and columns in a matrix is $\frac{M!N!}{2}$, which is computationally expensive. Thus, seriation in a matrix is performed heuristically.

Different matrix seriation methods use different objective functions to pursue heuristic solutions. For instance, Robinson [105] heuristically place the highest value along the diagonal in a matrix for seriation. The optimal leaf ordering method [8] begins with a hierarchical clustering of rows (or columns) and finds an order, which attempts to minimize the sum of distances between consecutive items in the dendrogram. Statistical analysis methods can also be applied for matrix seriation. For example, principal component analysis (PCA) [76] and correspondence analysis (CA) [55] treat a matrix as high dimensional data (rows as observations and columns as variables) and attempt to find two orthogonal axes as a 2D space to project such data. In this 2D space, the total variance of the data can be maximized, and the order on the two orthogonal axes is the seriation result. In this work, we use CA to perform seriation in lists for edge crossing reduction, and the detailed analysis about the connection between CA and edge crossings in lists is discussed in Section 5.2.1.
2.7 Max Entropy Principle based Bicluster Evaluation

The concept of entropy was originated from information theory [113]. In data mining, entropy measures how certain a model is about data. Lower entropy indicates that a model is more certain about data. Ideally, it would be perfect if we could infer a model that is quite certain about the data that we want to model. In this case, such a model would summarize the majority of the information contained in the data. However, in practice, the given prior information about the data is usually limited. To get a low entropy model, we may need to make additional assumptions about the data, which are not mentioned in the given information. Making such assumptions is unreasonable because we have no prior knowledge to support them. Moreover, the assumptions may not capture actual characteristics in the data. Thus, a reasonable choice is to avoid such assumptions and rely on the given information about the data, although this would increase the entropy of a model. This is what the Principle of Maximum Entropy addresses.

The Principle of Maximum Entropy (henceforth, MaxEnt) states that among all possible probability distributions, which are consistent with the given knowledge, the distribution with the maximum entropy best presents the current state of the knowledge about data and unbiased otherwise [71]. Here, these probability distributions are models of data that are inferred from the given information. It indicates that the distribution with maximum entropy, optimally uses the given information about data, and makes no additional assumptions for the unknown parts of data. This principle has drawn attention in the community of pattern
mining for subjectively surprising pattern discovery (e.g. [35, 127]). Here, surprising means that the piece of knowledge that will come next, is unexpected, based on the learnt knowledge.

The basic idea to measure surprisingness is that suppose we have a probability distribution $p$ that models the current beliefs about data. When evaluating data mining results, $p$ can be used to determine the likelihood of a result under the current knowledge of data. A high likelihood suggests that a mining result is probably known or can be inferred from the known knowledge. A low likelihood indicates that the mining results is surprising, which means it has much new information compared to the known part. Based on this, the MaxEnt principle has been used to evaluate the surprisingness of biclusters and chains [138]. This MaxEnt principle based evaluation can support exploring coordinated relationships by directing users towards promising biclusters or chains that can lead further analysis. In this work, we use it to prioritize biclusters and chains. The detailed rationale of this choice is discussed in Section 5.2.3.
Chapter 3

Five-level Design Framework

There are five levels of relationships (or connections) that underlie the notions of biclusters and chaining biclusters. These relations are closely related to the logic of the workflow that analysts may follow for sensemaking. To decompose the complexity of the discovery of relationships, these underlying relations are categorized into the following five levels (from low to high). Lower-level relations provide the critical basis that supports the exploration and identification of higher-level relations.

3.1 Five Levels of Relationships

**Entity Level: Single Entity Relationships (Entity-LR).** This is the most basic relationship, in the mathematical form of $1:1$. All other higher levels of relations build on this logical unit. In this relationship, for one entity in a particular domain, there is a correspond-
ing entity that comes from either the same domain or another domain that relates to this entity based on certain rules. For example, person A is related to city B, because person A has visited city B. Two domains, people and location, are involved in this relationship. As another example, gene X is similar to gene Y because they behave similarly under condition Q. In this relationship, despite the fact that there are two domains, genes and conditions, the two related entities are actually from the same domain, genes. To form relations in the next four levels, entities from different domains are necessary, so the discussion about Entity-LR refers to those with entities from two different domains, rather than the same domain.

**Group Level: Entity Group Relationships (Group-LR).** This level of relationship is in the mathematical form of $1:n$ or $n:1$. In such a relationship, there are, in total, $n+1$ entities from two different domains. The semantics of such a relationship is that for an entity in one domain, there is a corresponding related group of entities in the other domain. For example, 15 people are related to Amazon, because they all usually buy items from there or they all work for Amazon. A Group-LR relationship can result from the union of several Entity-LR relations that share connections with the same entity.

**Bicluster Level: Coordinated Relationships (Bicluster-LR).** This type of relationship is in the form of $m:n$. There are two domains with $m+n$ entities involved in this relationship, indicating that for a group of entities in one domain, a corresponding group of entities from another domain are related to them. For example, six people are connected with five locations, because they have each visited all the five locations. This level is represented by biclusters. This type of relationship can be formed by combining a series of
Group-LR where every single entity belongs to the same domain and the corresponding groups of entities in these different Group-LR relations are the same.

**Chain Level: Chained Coordinated Relationships (Chain-LR).** This is a more complex level of coordinated relations, in the mathematical form of \( m:n: \ldots :z \). Chain-LR can be considered an extension of Bicluster-LR because multiple individual coordinated-relations are connected together based on the shared entities between each pair. With intermediate groups of entities, at least three domains with \( m+n+z \) entities are connected with each other in Chain-LR. For example, four students, five cities and seven dates could be connected because all the four students visited the same five cities during the same week. Since there are more than two domains involved in this relationship, compared with Bicluster-LR, it takes more effort to mine or identify Chain-LR. It is also more difficult for humans to understand them, especially when the number of involved domains is large. However, Chain-LR contains more relations, which may provide analysts with meaningful story-like information (e.g., who plans to do what at which locations on what dates) for making hypotheses.

**Schema Level: Schema Level Relationships (Schema-LR).** This type of relationship presents highly abstracted, database-like, patterns within a dataset. Schema-LR indicates connections among all domains within a given dataset, which reveals an overview of the dataset. For example, in an intelligence analysis task, Schema-LR may refer to relations across all potentially meaningful domains for this task, such as people, organizations, locations, dates, and so on. Relevant domains within Schema-LR are usually defined or identified by domain experts, although some software (e.g., Entity Workspace [13], Jigsaw [117] and
Table 3.1: A Brief Summary of the Five-Level Design Framework.

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Format</th>
<th>Number of Domains</th>
<th>Number of Entities</th>
<th>User-Controllable Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Level</td>
<td>1:1</td>
<td>2</td>
<td>2</td>
<td>Domain</td>
</tr>
<tr>
<td>Group Level</td>
<td>1:n or n:1</td>
<td>2</td>
<td>n + 1</td>
<td>The size of a group; domain</td>
</tr>
<tr>
<td>Bicluster Level</td>
<td>m:n</td>
<td>2</td>
<td>m + n</td>
<td>The size of a bicluster; domain</td>
</tr>
<tr>
<td>Chain Level</td>
<td>m:n:...:z</td>
<td>At least 3</td>
<td>m + n + ... + z</td>
<td>The size of overlap between two biclusters; domain</td>
</tr>
<tr>
<td>Schema Level</td>
<td>1:...:1</td>
<td>Multiple</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

NetLens [78]) allow users to choose domains (from those that can be identified) based on specific tasks. Schema-LR can also be potentially formed on the basis of the search for involved domains by traversing those in all discovered Chain-LR.

Table 3.1 briefly summarizes these five levels of relationships. These relations cover most meaningful relations that analysts may want to explore, which leads to two design concerns: 1) how to visually represent these relations, and 2) how to interact with visual metaphors that can assist analysts to pick or find meaningful ones. Also, to enable human-in-the-loop [31] analysis, users may need control some key parameters in the data mining algorithms (e.g., biclustering and chaining), so that meaningful visualizations can be generated based on expected mining results. For Bicluster-LR, the size of a bicluster (the number of rows and columns) and domains are two key parameters for users to control; and for Chain-LR, the size of an overlap between two biclusters and domains of this share region are two user customizable parameters. Domains are also a user controllable parameter for both Entity-LR and Group-LR. The size of a group is another parameter for users to choose in instances
of Group-LR. There is no obvious user customizable parameters for Schema-LR because this relationship is usually determined by datasets. These parameters offer opportunities for interactions in bicluster visualizations.

3.2 The Five-level Design Framework

Exploration and identification of meaningful five-level relations are essential tasks for sense-making, which needs support from bicluster visualizations. On the basis of these relations, our discussion about the design framework of bicluster visualizations focuses on visual representation design and interaction design. The former addresses visual design choices for the five levels of relations with the purpose of summarizing feasible visual representation techniques to improve the perceptibility of computational results, especially biclusters and chaining biclusters. The latter discusses interaction design options with a principal task-driven purpose: guiding users to explore potentially meaningful relations. Thus, the interaction design can reinforce the perceptibility of visual representations by making them usable. With the combination of both aspects, we present a five-level design framework for bicluster visualizations to provide systematic design guidelines that inform the design of future visual analytics tools with use biclusters.
3.2.1 Design Choices for Entity and Group Levels

Several visual representations for graph layouts and interaction techniques have been discussed in [131], many of which can potentially be applied to present and explore Entity-LR and Group-LR. Entity-LR are easy for humans to interpret. Based on Entity-LR, Group-LR can also be easily formed given our previous discussion. In this section, my discussion focuses on node-link diagrams, since other visual representations (e.g., matrix) are more powerful to present higher levels of relations.

The node-link diagram is an intuitive way to visually represent relations between entities for relatively small datasets [60], although the shape of nodes or the type of links may be different (e.g., use circles or squares for nodes and use straight lines or curve lines for links). A single instance of Entity-LR has just two entities, and whatever shapes of nodes or types of links are used, it is easy for people to understand. By visually following an edge, regardless of its line types, people can easily understand that two nodes are related with each other. However, the situation becomes different when many Entity-LR instances, which may form Group-LR, are to be visualized, because there may be too many lines crossing with each other that obscures relationships among entities. The study from Ghoniem et al. [50] shows that there is significant difference in the node-link graph readability between the graph with straight lines and that with curved lines because curved lines are more efficient to reduce edge-crossing than straight lines. In addition, edge aggregation techniques, such as edge bundling [65], provide solutions to avoid clutter caused by too many lines. Selecting and highlighting (e.g., via changing shapes, colors, or size) are two important interactions usually applied in node-
link diagrams to help discriminate some relations from others, because highlighted nodes or links become visually prominent for humans to perceive. For example, a node with bigger size is easily differentiated from those with normal size. However, node-link diagrams can hardly represent *Bicluster-LR* and *Chain-LR* in an easily perceptible way due to the following three limitations:

**L1: Random locations.** In a node-link diagram, entities are randomly placed in the space by connecting one another with links. Without clearly visual, spatial structures, users have to manually reorganize the location of entities to form a new visual structure of the identified *Bicluster-LR* and *Chain-LR* so that they can easily understand.

**L2: Number of links does not scale.** Visually following links is the only way to explore relations between entities. The difficulty of doing so depends much on the number of edges in the graph.

**L3: Difficult to incorporate domain information to spatially aggregated entities.** Color coding is usually applied in node-link diagrams to indicate entities’ domain (or categorical) information. As a result, entities with the same domain information are not spatially aggregated, and users have to track both colors and links to find out two specific groups of entities that are related.

Using better layout techniques, matrix-based visualizations and parallel coordinates [69] are solutions that can help to overcome the above three limitations. Matrix-based visualizations and parallel coordinates show greater suitability for exploring *Bicluster-LR* and *Chain-LR*
than Entity-LR and Group-LR, so they are discussed later in Section 3.2.2 and Section 3.2.2, respectively.

Tree visualizations and layouts incorporating spatial distance (e.g. a force-directed layout [45]) are two common layouts for node-link diagrams. They can improve readability of the node-link diagram by overcoming $L1$ because the location of nodes is determined based on certain rules. For example, in a force-directed layout, two nodes are placed near each other because they are considered as similar. If instances of Entity-LR and Group-LR are in hierarchical relationships, tree visualizations are good choices. However, tree visualizations cannot be applied to explore Bicluster-LR and Chain-LR. The definition of a tree violates that of Bicluster-LR and Chain-LR discussed in Section 3.1, since all nodes in a tree belong to the same domain, rather than different ones.

When using spatial distance to enhance node-link diagrams, interactions that support spatially organizing information (e.g., dragging entities and spatially grouping entities) are key design concerns, which enable users to navigate and/or create spatializations for spatial reasoning [18]. Tools such as IN-SPIRE [136] and ForceSPIRE [38] implemented these design choices to support users to spatially organize visual metaphor of documents in the workspace. Vizster [58] applied the force-directed layout in the node-link diagram for social network analysis, and it used “blobs” (transparent coloring regions) surrounding entities to represent community structures. Noack’s LinLog energy model [92] applies energy-based and force-directed methods to layout clusters. Clusters in LinLog are defined as a group of nodes that have many internal edges but few external edges to nodes outside this group.
Similar to Vizster, LinLog uses spatial separation to show different clusters, but it does not show edges between entities. Because of missing edges, it is impossible to explore Entity-LR and Group-LR from LinLog’s visual representation without any necessary interaction (e.g., clicking nodes to show its edges). Spatial distance is readily perceived by humans, which can be used to indicate structures of a dataset. However, to find meaningful relations between specific entities, visual metaphors that work as scaffolding are still indispensable. Therefore, the number of links for entities still is a key constraint for the application of visually using spatial distance to explore Bicluster-LR and Chain-LR.

3.2.2 Design Choices for Bicluster Level

There are two major design concerns for Bicluster-LR: how to visualize a single bicluster; and how to visualize all possible biclusters identified from a dataset and navigate users to find meaningful ones. The first concern may lead to a simple visual metaphor of a single bicluster that is easy for humans to perceive, and the second concern may result in specific visualization techniques (e.g., focus+context [62, 84, 122]) that allows users to explore meaningful biclusters based on the context. Matrix-based visualizations and parallel coordinates provide possible visual solutions to meet these two design concerns, and the former has been studied in the bioinformatics domain [99, 109].
Matrix-Based Visualizations

Matrix-based visualizations represent *Entity-LR*, *Group-LR* and *Bicluster-LR*, where a relationship is indicated by a cell in the matrix and the two corresponding entities are respectively listed as a row name and a column name of the matrix. For *Entity-LR* and *Group-LR*, compared with node-link diagrams, matrix-based visualizations are less intuitive for humans to perceive [51]. For *Bicluster-LR*, matrix-based visualizations are superior to node-link diagrams by overcoming the three constrains mentioned in Section 3.2.1. In a matrix, entities are listed as names of rows or columns, rather than randomly located in the space. By using cells to indicate relations, matrix-based visualizations effectively avoid visual clutter [107] caused by edges crossing and/or overlapping. Besides, domain information can be easily incorporated into matrix-based visualizations, and entities fitting in the same domain can be spatially listed near each other. For example, columns and rows of a matrix respectively belong to two different domains. This offers a clear visual representation for a single bicluster.

Bixplorer applied this idea to visualize individual biclusters mined from textual datasets, and reported that users could perform text analysis using these visual biclusters [43]. However, for an overview of all mined biclusters from the text, Bixplorer simply listed all mined biclusters, requiring users to select biclusters from the list and view them in a detailed preview panel to determine whether each bicluster might be useful. Bixplorer emphasized a bottom-up approach, enabling users to discover relevant biclusters based on the documents and entities in their focus of investigation. The relevant biclusters were visually embedded directly into
a user’s spatial document workspace, thus placing them in context. Methods are needed to enable top-down overview of textual datasets from the perspective of biclusters that help direct users to meaningful biclusters and then to supporting details in documents.

Similar to Bixplorer, matrix-based visualizations enhanced with heatmaps are widespread in the bioinformatics domain for gene expression data analysis (e.g., BicAT [9], BiCluster Viewer [59], BicOverlapper 2.0 [111], BiGGEsTS [52], BiVoc [56], Expression Profiler [80] and GAP [139]). To perform gene expression analysis, the collected raw microarray data are transformed into gene-expression matrices, where rows usually represent genes and columns stand for conditions [17]. Matrix-based visualizations are a good fit for this task. By simultaneously reordering rows and columns in the matrix, biclusters can be formed from the gene-expression matrices [89, 126], which helps to identify co-expressed genes under a shared set of conditions. Les Misérables Co-occurrence\(^1\) developed with D3 [15] is a good example that visually shows this process. Compared with static visualizations, presenting the dynamic reordering process helps users to understand how biclustering works and how biclusters are formed from a matrix.

A typical matrix-based visualization that shows the result of two biclusters identified from gene-expression matrices is shown in Figure 3.1\(^2\). The big matrix represents a gene-expression matrix and two small matrices indicate two identified biclusters. Comparing the two biclusters, we find that all genes are the same except two, and there are six conditions shared across

\(^1\)This visualization can be found at http://bost.ocks.org/mike/miserables/

\(^2\)Taken from http://genomics10.bu.edu/terrence/gems/help.html
Figure 3.1: An example of the matrix-based visualization to illustrate two biclusters mined from a gene-expression matrix.

the two biclusters. Parts of relations within the two biclusters overlap, so it is impossible to visually separate these two biclusters by just reordering rows and columns of the big matrix. Therefore, although the big matrix contains all relations to form biclusters, extra techniques are required to layout all possible biclusters in a human perceptible way and navigate for exploratory analysis. This is conceptually a double Euler diagram problem on two domains simultaneously.

Grothaus et al. [56] proposed an automatic layout algorithm that allows for replicating rows or columns to optimize the layout of matrix-based bicluster visualizations. The optimization refers to two aspects for the big matrix: 1) to form contiguous subregions and each of them contains as many overlapping biclusters as possible; and 2) to keep the size of the big matrix
as small as possible. Bicluster Viewer [59] applied this algorithm to a matrix-based visualization with five key interactions to help navigate and explore biclusters from gene-expression matrices. Users can zoom in/out of the matrix, and highlight selected biclusters and their corresponding rows and columns. Users can choose to show all biclusters within the big matrix, with replicated rows and columns, and a rectangle with a colored frame is used to indicate the region of each bicluster. In addition, Bicluster Viewer can show biclusters without replicating any rows or columns. In this case, each bicluster may be split into different subregions within the big matrix, which are visually indicated by rectangles with the same colored dashed lines. To help navigate among biclusters, Bicluster Viewer maintains a list of identified biclusters, where the selected biclusters are highlighted with yellow and biclusters formed by replicating rows and columns are colored with red. However, similar to the problem with Bixplorer, the bicluster navigation list in Bicluster Viewer displays all biclusters in a simple list without user-defined names or labels. The arbitrary bicluster identifier names do not provide users with semantic information. Semantic meanings influence human interpretation [21, 40], which requires appropriate information scent to enable users to understand relations within a bicluster in a brief manner.

A matrix-based visualization provides an efficient visual representation for a single bicluster, which is easy for human to perceive. By replicating rows or columns, it is possible to layout all biclusters within a big matrix, and interactions applied in Bicluster Viewer provide a feasible instance to help users navigate among these biclusters. However, these replicated rows or columns may cause confusion, particularly when they are repeated several times and
these repetitions may appear spatially near or far from each each. Given the combinatorial nature of biclusters, this process may require a large number of row and column replications.

Reduced Parallel Coordinates with Two Domains

Parallel coordinates is a well explored visual technique to present high-dimensional or multivariate data [64, 97, 134]. Since Bicluster-LR just has two involved domains, our discussion of parallel coordinates in this section refers to the reduced version with two domains. Instead of randomly placing entities in a two-dimensional space, parallel coordinates spatially sorts entities in a list based on their domains. Compared with the node-link diagram discussed in Section 3.2.1, parallel coordinates uses locations to separate one group of entities from another, and display them in an easily perceivable way. For example, in parallel coordinates, Entity-LR is represented as two entities from two lists with a line between them; Group-LR is displayed as one entity from a list that has several lines connecting with several entities from another list, and Bicluster-LR is more complex, which is represented as many entities in a list and each of them has the same number of links to the same entities in another list.

Jigsaw applied parallel coordinates in its List View [117], where entities are organized in different lists based on their domains. Similarly to [75] and [42], Jigsaw allows users to select domains (e.g., people, dates, locations, etc.) to be displayed in the List View. With interactions such as selecting, highlighting and ordering entities, users can easily explore Entity-LR and Group-LR of interest in Jigsaw. With these interactions, analysts can find biclusters in Jigsaw, and an example is shown in Figure 3.2. In this example, The Sign of the Crescent
Figure 3.2: An example of finding a bicluster in Jigsaw’s List View. Yellow indicates entities that a user selected and orange indicates relevant entities corresponding to the selected one(s) based on word co-occurrence.

dataset [68] is imported in Jigsaw and it takes three steps (from A to C) to find a bicluster that indicates the three key persons involved in the Atlanta event and a group of locations that they all visited. Jigsaw uses highlighting, particularly highlighting relevant entities by orange based on word co-occurrence, to guide users to perform exploratory analysis. However, there is little guidance for users to find biclusters. After step A in Figure 3.2, how do users know which entity in the list to consider adding into the bicluster in the next step?

---

3Version 0.53, from http://www.cc.gatech.edu/gvu/ii/jigsaw/
Users have to apply trial-end-error to finally reach the 3x6 bicluster shown in Figure 3.2. Thus, visually discriminating entities in the same list may better help analysts to find entities for their next analysis step. Coloring relevant entities in the same list based on the number of entities (in another list) shared with the selected one is a possible solution.

Parallel coordinates is also applied in bioinformatics to display biclusters (e.g., BicAT [9], BiCluster Viewer [59], BicOverlapper 2.0 [111] and BiVisu [23]), where each vertical axis indicates a condition and a polyline represents a gene. However, compared with matrix-based visualizations, parallel coordinates is less used in this domain [109] and few interactions are available in these tools. Parallel coordinates may do a better job to show variation in trends of genes under different conditions than explore biclusters. Although these tools show results of possible biclusters, none of them perform user studies to evaluate whether these biclusters in parallel coordinates can be perceived or not. Johansson et al. [74] tested readability of parallel coordinates with five stimulus patterns, and their study shows that difficulty to discriminate these five patterns in parallel coordinates increases when the noise level goes above 13%. This suggests that if lines indicating relations of different biclusters can be visually well organized, users may identify biclusters in parallel coordinates.

**Four Design Choices.** There are four different design choices discussed in relevant literature that can be applied to improve the display of biclusters in parallel coordinates. The most basic one is to move entities belonging to a bicluster together and use different color to highlight them, such as step D in Figure 3.2. This aggregates entities spatially close with each other, which helps to separate these entities from others. Another way is to replace
straight polylines with curved lines [54] and add force, similar to force directed layout, to these curved lines to aggregate or bundle them based on certain rules [148]. In this way, possible biclusters can be explored by starting analysis from the aggregated curved lines. The third way is to aggregate entities and polylines respectively and use colored ribbons, similar to the Bubble Sets technique [29], to wrap the aggregated entities and polylines from the first vertical axis to the final one [6, 46, 81, 85, 96, 147], which can be used to indicate a bicluster that is determined by the smallest set of the shared entities across all axes in this region. Finally, tile-based parallel coordinates [3] provides an efficient way to avoid visual clutter, since it divides the plotting space into rectangular tiles and colors these tiles based on the sum of polylines that intersect with the tile. This can be applied to show biclusters with a modification of color coding rules for tiles. For example, based on the selected entities, a set of polylines (denoted \textit{SetA}) can be formed by a union operation of all polylines starting from these entities. Then for each tile, a set of shared polylines (denoted \textit{SetB}) can be found by an intersection operation between the polylines passing through this tile and those in \textit{SetA}. Finally, each tile is colored based on the total number of polylines in \textit{SetB}. By following the colored tiles, it is possible to identify whether biclusters exist or not for the selected entities.

These four design concerns provide possible solutions for presenting a relatively small number of biclusters in parallel coordinates. If there are overlaps between biclusters, the first design choice will not completely display all biclusters unless some entities are replicated. Applying the third design choice with replicated entities, biclusters in parallel coordinates are still
difficult to identify, because many regions may overlap with each other. These overlaps may lead to misunderstanding and obscure the exact number of clusters [33]. The second design choice avoids aggregating entities together, but visual clutter may still appear due to many curved lines, especially when there are many biclusters. For the last design choice, if many biclusters exist, various tiles may have the same color. In this case, it is difficult to differentiate biclusters. However, the 1-dimensional sorting of parallel coordinates should reduce the replication problem in comparison to the 2-dimensional sorting of the matrix-based approach. Although there may be some interesting optimizations that attempt to sort two vertical axes in parallel coordinates so that the bicluster links are as horizontal as possible. Ultimately though, this solution devolves into a linear list of biclusters as used in Bixplorer.

![Figure 3.3: An example of rearranging axes by switching axes BB and CC.](image)

To overcome these drawbacks, interactions are a key requisite to identify meaningful biclusters. In addition to the basic interactions mentioned in the Jigsaw example (e.g., select and highlight entities), brushing and rearranging axes are two important interactions to explore parallel coordinates [116]. Brushing allows users to create a customized region (e.g., a focus
area) in an axis and move it to select a set of polylines [118]. By following these polylines, analysts can determine whether biclusters exists or not for entities enclosed in the bin. For example, all three entities shown in step D in Figure 3.2 have six lines connecting to the same six locations, so that the three people and the six locations form a bicluster. Axes rearrangement assists to explore relations between two specific axes, which may reduce polylines crossing and help to find relations between entities in two nonadjacent axes. An example of axes rearrangement in parallel coordinates is show in Figure 3.34. By switching two axes BB and CC, relations between entities in AA and CC are clearly revealed. If axes AA, BB and CC are different domains, axes rearrangement also provides a feasible way to explore Chain-LR.

Zoned Node-Link Diagram

Node-link diagrams can also be used to explore Bicluster-LR. BicOverlap [110] applied modified node-link diagrams to show biclusters and overlaps among biclusters. In BicOverlap, each node represents an entity, and the layout of nodes are determined based on a force-directed layout algorithm. Nodes in different domains are indicated with different visual marks. All nodes in each bicluster are wrapped in a “zone”. The boundary of this “zone” is determined by the outermost nodes. To avoid visual clutter, edges between each pair of nodes are hidden. It seems that this design works for both single and all biclusters cases, but at least three drawbacks exist. Visual marks help to discriminate one domain from another,

---

4Modified based on the example of reordering from http://syntagmatic.github.io/parallel-coordinates/
but they are hard to remember without a legend. By hiding edges, the visual representation in BicOverlap implicitly emphasizes entities rather than relations, so relations may be obscured by a large number of entities. Overlaps among “zones” indicate overlaps among biclusters, but the perceptibility of this depends on the number of biclusters overlapping with each other. Small-size biclusters, those with a small number of entities, within a heavily overlapping region may be ignored. However, the advantage is that this design is able to convey an overview of biclusters within a dataset. Furthermore, enhanced with interactions (e.g., filter biclusters by domains and popup a bicluster of interest), this design can help explore meaningful biclusters within the overall context.

3.2.3 Design Choices for Chain Level

How to visually represent a bicluster-chain or all bicluster-chains and help users navigate to meaningful chains is a crucial task for Chain-LR visualizations. Hybrid matrix diagrams provide a feasible solution to fulfill these demands. Hybrid matrix diagrams combine node-link diagrams or parallel coordinates with matrix based visualizations, which substitutes nodes in the node-link diagram or entities in axes of the parallel coordinates with matrices. Parallel coordinates discussed in this section are those with multiple domains and design concerns discussed in previous section can also be applied here. Each matrix in the hybrid diagram indicates a bicluster, and the node-link diagram or parallel coordinates illustrates how several biclusters are connected together, which also specifies the structure of bicluster-chains.
Node-Link Diagrams + Matrices

NodeTrix [61] and Bixplorer [43] are two systems that apply hybrid matrix diagrams, and a similar design is also mentioned in [10]. An example of hybrid matrix diagrams generated with Bixplorer⁵ is shown in Figure 3.4. In this example, there are three biclusters that (from left to right) respectively represent relations between people and location, phone number and people, and date and phone number. Curved lines indicate shared entities between two biclusters. Bixplorer’s bottom-up approach allows users to interactively expand chains from a given bicluster.

![Figure 3.4: An example of bicluster chain with three biclusters in Bixplorer.](image)

In a more top-down design that attempts to support Ben Shneiderman’s visual information seeking mantra [114] of overview first, NodeTrix uses a node-link diagram to show many connected matrices. NodeTrix provides three types of links: “underlying links” (simple

⁵The tool can be found at http://recsys.cs.vt.edu/mineviz
curved lines, same as those in Bixplorer), “underlying links with full size” (curved lines, the thickness of which equals to the width of a matrix cell’s edge) and “underlying links with attributes” (curved lines highlighted with different colors). The first type of link shows detailed connections, and the last type of link visually differentiates some links from others. Users can also extract a node from a matrix and merge two matrices together in NodeTrix. 

The design of NodeTrix is able to show several bicluster chains, actually a graph of related biclusters. The interaction design of NodeTrix concentrates on assisting the analysis of the connected parts of the graph by splitting and merging biclusters to explore alternate configurations of chains. Still, there is a need for designs that can guide users in exploring these chains.

Visually dynamic path extraction [67] seems a promising way to enhance NodeTrix’s design for chain exploration. enRoute [98] implemented dynamic path extraction for biological pathway analysis, and used Bubble Sets [29] based techniques (using isocontours to create a colored region to wrap a set of entities) to visualize a selected path and its alternatives. In enRoute, all possible paths between the user-selected start and end node are visually presented, and users can add nodes to extend selected paths. Incorporating this into NodeTrix’s design, users can begin chain exploration by choosing a start and end bicluster, or the system shows all computed chains with heatmap styled bubble sets. The color of these bubble sets can be encoded based on the two domains of a bicluster. For biclusters with the same two domains, the more cells in a bicluster, the darker its color will be. Moreover, user-selected biclusters or chains can rise up to the front layer. Thus, based on this design,
users can perform chain analysis by starting with the overview and seeking guided visual metaphors on demands.

**Parallel Coordinates + Matrices**

Some design considerations discussed in *Section 3.2.2* (e.g., wrap entities and polylines with colored ribbons, brush and rearrange axes) also apply for *Chain-LR* exploration. The design of hybrid diagrams that combine parallel coordinates and matrices provides a better solution for chain exploration, and it was applied for comparing results of different clusters. HCE [112] applied parallel coordinates with matrices to compare results of two hierarchical clustering algorithms of genomic microarray data, and its biology users showed positive feedbacks about this design. The Caleydo Matchmaker technique [86] applied this design to conduct visual comparison among multiple groups of clusters. In this design, matrices in each axes can represent biclusters with two specific domains, and matrices connected among multiple axes can indicate chains. Compared with the previous hybrid diagram design, biclusters in this design are better organized. Since entities are aggregated into different biclusters, compared with parallel coordinates with entities on axes, this design reduces the number of links between each pair of adjacent axes. Moreover, using colored ribbons or the *Bubble Sets* technique to wrap matrices and links works as a salient visual representation of bicluster chains.
3.2.4 Design Choices for Schema Level

Schema-LR indicates relations among domains. The number of domains is much smaller than that of entities, so visual representations of Entity-LR can also be applied to Schema-LR. For example, the normal node-link diagram (e.g., database schema diagrams) is an obvious visualization that can clearly convey Schema-LR in a dataset. In this representation, each node stands for a domain in the dataset, and the thickness of links can indicate the strength that two domains are connected. The connection strength can be calculated based on the number of connections between entities in the two domains or the number of biclusters formed with entities of the two domains, or the number of chains participated in.

This design gives a clear overview of the dataset, and an alternative design is the clutter map proposed in [44]. A clutter map is similar to a node-link diagram with more detailed information. In the clutter map, the size of nodes is determined based on the number of entities belonging to this class (or domain) and edges are balloon-shaped that are visually merged with connected nodes. The size of the balloon depends on the shared entities between two domains, which can be modified to determine balloon size based on the number of biclusters relevant to the two domains. Another similarly applicable design is that from PivotGraph [133], which lays out aggregated nodes in a grid, similar to a chessboard. Node positions in the grid are determined by an algorithm that minimizes the number of edge-crossings. The size of nodes and the thickness of edges can be applied to encode the number of entities and biclusters respectively. Curved lines with arrowheads are used in PivotGraph to demonstrate directed edges, if there is any, which can also be applied to direct possible
analytical paths for users to explore. For example, the path from Domain A to Domain B, then to Domain C is thicker than the path from Domain A to Domain B, then to Domain D, which indicates that starting analysis with Domain A to Domain B and Domain C may be more reasonable.

A fourth design is the chord diagram\(^6\), which is inspired by Circos [82]. In the chord diagram, each chord can represent a domain in the dataset, and the length of a chord depends on the number of entities in this domain. Ribbons connecting two chords can indicate biclusters relevant to the two domains, and the thickness of ribbons may be determined by the number of the shared biclusters. Compared with the previous three design options, the chord diagram may not work well for a dataset with too many domains, because all chords are aligned in a circle. If the number of domains is large, the length of each chord will become small, and the number of ribbons will grow, which may lead to visual clutter inside the circle.

An important task of Schema-LR visualizations is to direct users to drill down to one or several domain(s) to explore more information for further analysis. On the basis of understanding visual representations of Schema-LR, users still need interactions to find domains that are meaningful for them. Dynamic path extraction, as discussed in Section 3.2.3, presents patterns among different domains based on extracted paths. This may guide users to further explorations of bicluster-chains related to some specific domains. It may also be useful to consider a chain-centric approach to Schema visualization, which overlays many chain paths onto the schema diagram.

\(^6\)An interactive example is at http://bl.ocks.org/mbostock/4062006
3.3 Four-Level of Interaction Design

From the perspective of intent, there are four-level of interactions that potentially can be applied to bicluster visualizations: **Readability Level** (Readability-LI), **Navigation Level** (Navigation-LI), **Parameter Level** (Parameter-LI) and **Object Level** (Object-LI) [39]. Readability-LI aims at improving the readability of visualizations, so most interactions discussed in Section 3.2 (e.g., select or highlight a node) belong to this category. Navigation-LI enables users to navigate between the five relationship levels and their visual representations. Parameter-LI enables users to control key parameters of algorithms. Object-LI helps users focus on their analytics process. Navigation-LI enhanced with Parameter-LI provides promising solutions to navigate visualizations of one relationship level to another.

For Parameter-LI, users are exposed to the underlying algorithm parameters. Using sliders to control the parameters values of an algorithm is a common example. For instance, iPCA [72] implemented this, where users can control how much a dimension will contribute to the PCA calculation by manipulating the sliders and choosing dimensions by check boxes. Bixplorer applied this to enable users to control the minimum size of biclusters to be identified from a textual dataset [43]. PivotSlice [145] allows users to drag parameters to specify domains for relationship discovery.

By controlling the parameters in the last column of Table 3.1, it is possible for users to navigate in either a top-down manner or a bottom-up one. For a top-down example, by choosing domains in Schema-LR, users can get specific Chain-LR or Bicluster-LR that they
then can drill down to the Group-LR and Entity-LR. Bixplorer applied this by enabling users to extract, from a bicluster, a row or a column as a thin bicluster, and further extracting a cell from the thin bicluster. The row or column belongs to Group-LR and the specific cell is an instance of Entity-LR. Conversely, in a bottom-up fashion, in Bixplorer users can also merge cells together to form a new row, a new column or a new bicluster in the form of a matrix, and then users can link their customized biclusters with each other to form a new chain that is meaningful for them. Thus, Parameter-LI of the higher level relations determines the computational results of the lower level, while Navigation-LI directs users to actually transform from one level of visualizations to another level. Visualizations at each of the five levels can be either linked or visually integrated to enable drill-down and roll-up through the five levels. Bixplorer visually integrates the lower levels, but also provides separate linked views (lists) of the higher levels.

Similar to Parameter-LI, Object-LI can be applied to enhance Navigation-LI between and to direct the mining of biclusters and chains on various domains. Object-LI is an implicit way to control algorithms compared with Parameter-LI, since not all users realize that some of their interactions with visual metaphors are used as parameters of algorithms to control the future output. ForceSPIRE [38] is an example of such interactions (e.g., moving, annotating and highlighting) that re-weight the term dimensions in the distance metric and recalculate similarities among documents. EvoGraphDice [16] employed a similar idea to dynamically change a scatterplot matrix. Object-LI enables users to focus on the exploratory analysis (for Bicluster-LR and Chain-LR) by implicitly tuning the algorithm parameters. For example,
in Analyst’s Workspace [67], the system shows bicluster chains to users that connect pairs of documents that the user interacts with, and then updates the chains by adding more relevant biclusters into the path of the chain based on the specific biclusters and documents that the users either keeps or eliminates. This implicitly incorporates analysts’ judgement about members of a bicluster chain into the computation or visualization of results.

3.4 Summary

All specific design choices discussed above are summarized in Figure 3.5. Node-link diagrams support tasks relevant to almost all five levels, although variation exists for each specific level. Matrix based visualizations and parallel coordinates are valuable for the exploration of Bicluster-LR and Chain-LR. The hybrid visualization that combines the node-link diagram or parallel coordinates with matrices can do a better job, because it can display the overview of a dataset with the structure from the node-link diagram or parallel coordinates, and detailed relations buried in matrices. Several tools implement these design choices and have been evaluated with user studies. Positive user feedback from them indicate that these visual representations are good directions to pursue. However, at least three challenges still exist when applying this design framework to the design of future visual analytics tools with biclusters.

C1: Integration Challenge. How to pick design options from this framework and snap them together into an integrated whole is a basic problem. Users likely need multiple coor-
ordinated visualizations for exploratory analysis across all five levels [95]. We have identified
design options for each level but still lack examples that successfully combine them together
to fulfill tasks across all five levels for sensemaking.

**C2: Traversal Challenge.** How visual analytics tools should guide users’ traversal through
this five level framework is still not clear. Although Shneiderman’s visual information seeking
mantra has much impact on visualization design, the analytical process may not always work
the same way. Biclusters offer a bridge connecting the overview and details in a dataset.
However, at which end should bicluster visualizations start (i.e., Schema-LR first, or Entity-
LR first)? How can we enable rapid bi-directional navigation among these levels, as suggested
by Pirolli’s sensemaking model [101]?

**C3: Layout Challenge.** How to effectively layout all biclusters and bicluster chains in an
overview still needs further research. Although possible solutions are discussed in this design
framework, the replicated information may still cause confusion. Also, it may be difficult to
enable users to customize the layouts generated by automatic layout algorithms.

These three challenges direct future research paths for the further exploration of a design
space of bicluster visualizations. **C1** brings the question of how these design choices can
be combined together. Results from **C2** may give clues to the answer to **C1**, because the
design of bicluster visualizations should follow users’ analytical processes. Together with
novel layouts identified in **C3**, this framework and agenda can make biclusters usable for
efficiently discovering coordinated relationships in visual analytics.
Figure 3.5: A summary of the five-level design framework for bicluster visualizations.
Chapter 4

BiSet

The five-level design framework for bicluster visualizations, discussed in previous chapter, is based on five hierarchical levels of relationships potentially existing in a dataset [125]. Relations in higher levels (e.g., bicluster-level and chain-level) are usually constructed based on those in lower levels (e.g., entity-level and group-level), so relations in lower levels provide a critical support for the exploration and interpretation of those in higher levels. These five levels of relations systematically present the space of relationship, which works as an important guideline for us to follow.

In this chapter, we present BiSet [123], a visual analytics technique to support interactive exploration of coordinated relationships with biclusters. In BiSet, coordinated relationships are formalized as biclusters and algorithmically mined from a dataset. Biclusters are visualized in context as bundled edges between sets of related entities. These bundles
enable analysts to infer semantic insights about potentially coordinated activities. Bundles are made as the first class objects and a new layer in-between lists is added to contain these bundle objects. Interactions are applied to both edge bundles and entities for revealing and organizing relevant information in a bi-directional way. Users can interact with edge bundles to forage and organize related entities and, vice versa, for sensemaking purposes.

4.1 Design Requirement Analysis

4.1.1 A Design Trade-off

![Diagram](image)

Figure 4.1: An example of a bicluster, indicating a coordinated relationship between three people and four locations. (A) presents all connections between each pair of related entities from the two domains. (B) shows the result of bundling edges in this bicluster. (C) demonstrates the result of both bundling edges and grouping entities in this bicluster.

Visually a bicluster can simultaneously bundle connections and group entities and this can be clearly conveyed in a list view, shown in Figure 4.1. However, cases are not always that
easy. When certain entities belong to more than one bicluster, it becomes difficult to visually group all entities of the same bicluster together. Figure 4.2 shows an example of this case. There are three biclusters indicating three different coordinated relations between people and locations. They share some entities (e.g., P1 and P2 are associated with both bicluster A and bicluster B). This brings about an Euler diagram problem [4, 106] when visually grouping entities. When the number of shared entities increases, it becomes more difficult to present a visual representation that show the membership of these entities without replicating some of them. This suggests a key design trade-off: entity-centric versus relationship-centric, which means that we cannot easily achieve the goal of clearly presenting both entities (without duplications) and relationships (without separations). Techniques such as bubble sets [29] and untangling Euler diagrams [104] attempt to balance this trade-off by using a 2D space. They show relationships with their members in a calculated spatial layout with certain boundaries. However, they do not scale up well. In a list, there is just one dimension to use for organizing entities, so it is even harder to balance this trade-off.

**Entity-centric** requires that entities that belong to a certain domain (e.g., people in Figure 4.2) should be listed in a certain order without duplication. The positions of entities can be reordered to fulfill some purpose (e.g., listing names in an alphabetical order or ranking them based on frequency in documents). Since entities cannot be duplicated, relationships that consist of shared entities may be separated apart. Thus, an entity-centric design can help to avoid the ambiguity caused by entity replication, but it costs the completeness of relationships.
Figure 4.2: A detailed example to illustrate the Euler diagram problem that arises when visualizing the membership of entities in the domain of people shared by three biclusters. This problem indicates a key design trade-off: *entity-centric* versus *relationship-centric*.

**Relationship-centric** requires that entities that belong to the same relationships (e.g., biclusters) should be placed near each other, which visually preserves the completeness of relationships. To achieve this, entities may be duplicated, particularly when several relationships share two or more entities. A relationship-centric design (e.g., bubble sets technique [29]) can maintain the integrity of relationships with the cost of entity duplication. This may confuse users, especially when they see some entities appear several times at several different positions in a list.
4.1.2 Layout Candidate Selection

Three layouts can be potentially applied to show coordinated relationships: node-link diagram, matrix, and list (including parallel coordinates). A detailed discussion about applying them in bicluster visualizations are addressed in [125]. Because of the following two advantages, we choose list as the major layout of BiSet.

**Consistent, domain-based entity management.** In a list layout, entities are organized in a consistent manner by domains. In a node-link diagram, nodes are placed either randomly or based on certain layout algorithms (e.g., force-directed layout [45]), so it is hard to separate entities of one domain (e.g., people) from those of another (e.g., location). Compared with a node-link diagram, a matrix is a better organized layout, where relations are more readable [50]. Entities in a matrix are organized in two orthogonal directions based on domains. However, when a new group of entities are to be added, it is impossible to add them in the existing matrix. A new matrix has to be built to show connections between the newly added entities and those from one domain of the existing matrix. This leads to two problems: entity duplication and direction (row or column) selection. To build the new matrix, entities from one domain of the existing matrix have to be duplicated to form either row names or column names in the new matrix. If replicated entities work as the row names in the new matrix, newly added ones will be the column names, and vice versa. Compared with matrices, in a list whenever a group of entities are to be added or removed, we can easily add or remove one list. Thus, compared with the other two layouts, lists organize entities in a consistent manner.
Organized alternate subspaces for entities and relationships. In a list view, a 2D space is sliced into two types of subspaces, where entities and relationships appear alternately. Figure 4.3 gives an example of such slicing that generates three subspaces for entities and two subspaces for relationships. This provides an opportunity to leverage the design trade-off by using the relationship subspace. In node-link diagrams, the space of entities is intertwined with that of relations so there is no clear boundary between the two spaces. Matrices potentially emphasize the space of relationships (e.g., the structure of a dataset) [50], so it is difficult to support simple relationship (e.g., entity-level or group-level relations) exploration in a matrix. Visually, in a matrix, the proportion of the space for relations (the total area of all cells in a matrix) is larger than that for entities (one column and one row in the matrix) and the ratio of the two increases when the size of a matrix gets larger. Thus, compared with the other two layouts, lists slice a 2D space for entities and relations in a clearly organized and usable way.
4.1.3 Requirements Description with Identified Tasks

Based on such design trade-off and selected layout, four important design requirements are identified as follows.

**R1: Entity and relationship encodings.** Efficient visual encodings for both entity and relationship are necessary which should attempt to achieve four important goals. First, visual encodings should assist users to discriminate entities from relations (a). Then visual encodings (particularly for relationships) should potentially help to reduce visual clutter (b). Third, visual encodings should provide clues to reveal the membership of entities (c). Finally, visual encodings should reflect the changes when the state of entities or relations updates (d).

**R2: Four types of exploration.** There are four possible types of exploration in a list layout based on the space slicing: i) from entity to entity, ii) from entity to relationship, iii) from relationship to relationship, and iv) from relationship to entity. The first type of exploration refers to when users start from entities and focus on finding related entities. The second one may happen in the scenario where users take the strategy of following the clues from identified meaningful entities [79] and search for more relevant information. The third type of exploration may be performed when users seek additional relationships based on current one. This is a possible case for text analytics which has been reported in [121]. The last one may be used to compare several hypotheses, which has been identified as a common intelligence analysis strategy [26]. For example, several biclusters share some individual relations, but
they still have their unique ones. This may form (partially) conflicted relations that lead to different hypotheses. In this case, users may have to compare these biclusters by drilling down to detailed level of information (e.g., entities) for competing them.

Moreover, there are two ways to conduct the four types of exploration: from one to many and from many to many. For example, users may want to find several relevant entities based on one or multiple biclusters. Based on this, we identify eight user tasks involved in such explorations, which are summarized in Table 4.1. For each type of exploration, users can start from either an entity (or a bicluster) or multiple entities (or biclusters) and then try to find relevant entities or biclusters. A user’s analytical process may consist of a series of these tasks that iteratively forage relevant information and identify some meaningful pieces [79].

**R3: Organizing entities and relationships.** Entities and relationships should be visually represented in an organized way. This can help users to easily find useful information.
In addition, users may want to make changes to the automatically generated layouts so that they can organize entities or relationships in personalized, meaningful ways (e.g., using spatialization) for sensemaking [5].

**R4: Retrieving original data for reference.** To evaluate algorithmically discovered coordinated relationships, users may refer to the content from the original dataset (e.g., documents) because they need contextual information to help them to interpret and further evaluate these relations [53]. BiSet should attempt to efficiently direct users to useful information, rather than keep them from reading documents.

### 4.2 BiSet Technique

Three key aspects are involved in BiSet: coordinated relationship discovery in data level; *bundles as objects* and a “in-between” layer in visual level; and interactions to support four types of exploration and two ways of organizing information. In this section, we discuss them in detail and explain how identified design requirements are satisfied.

#### 4.2.1 Data Level: Bicluster Discovery

Coordinated relationship discovery is the fundamental step in BiSet since it determines how edges are bundled. In BiSet, we formalize coordinated relationships as biclusters. Suppose that entities have been extracted from a dataset (e.g., documents) with named entity
recognizers such as LingPipe [19] or similar tools. We use closed itemset algorithms (e.g., LCM [129] and CHARM [141]) to discover biclusters based on extracted entities. Each unique pair of entity types (e.g., people and location, people and date, location and date, etc.) is considered a type of coordinated relationship and is computed separately to generate results that include all unique pairs of entity types. Results are stored in a database and associated with the dataset under investigation.

The mined biclusters indicate different coordinated relationships and some of them may share entities and relations in entity-level or group-level. This suggests that some entities and edges (individual relationships) are members of biclusters. Thus, membership in BiSet, in the data level, can be considered from two aspects: entity and edge.

4.2.2 Visual Level: Bundles as Objects and “In-between”

In BiSet, we propose two important concepts to balance the key design trade-off: making bundles as first class objects and adding a new layer “in-between” lists to contain bundle objects. The former enables users to directly manipulate relationships (relationship-centric) and the latter helps to visually reveal membership of entities in two neighboring lists without duplicating entities (entity-centric).

Making Bundles as Objects  In BiSet, we make bundles the first class objects so users can directly manipulate them for sensemaking purposes (e.g., organizing information). BiSet bundles edges based on computed biclusters that reflect algorithmically discovered coordi-
nated relationships. Different from spatial edge bundling techniques that emphasize bundling based on spatial proximity, BiSet bundles edges based on coordinated relationships that reveal *task-oriented semantic insights*. This assures that *edge bundles remain stable*, regardless of the positions of associated entities. Thus, bundles potentially enables users to use space (e.g., vertical position) to organize information (e.g., entities) (for *R3*), and safely retrieve related information by interacting with edge bundles (for *R2, iii* and *iv*).

**Adding a *in-between* Layer**  To make bundles usable, we add a new layer, called “in-between”, visually locating in the space between two neighboring entity-lists. It contains bundles and edges (e.g., those that do not belong to any coordinated relationship). In this layer, BiSet allows users to manipulate bundles for sensemaking (e.g., organizing entities and checking their membership), so bundles can support users interactively exploring coordinated relationships.

**Semantic Edge Bundling in BiSet**

BiSet has two types of edges: *independent* and *associated*, which are mutually exclusive. The former refers to edges that do not belong to any coordinated relationship and the latter are those that can form one or more coordinated relationships. For instance, in Figure 4.4, the edge on top in (A) is an independent edge and other edges are associated ones. Independent edges can reflect *entity-level* and *group-level* relationships, but they are not associated with others to form coordinated relationship. Based on membership, associated edges that belong
to the same bicluster can be aggregated and represented as an edge bundle. BiSet takes the following three steps to bundle edges (for $R1(b)$).

1) *Grouping edges based on membership.* We separate associated edges into different groups based on their associated biclusters. For those in multiple biclusters, we duplicate and assign them respectively to multiple groups, so each group has a complete number of edges.

2) *Bundling edges based on groups.* For each group obtained from the previous step, we bundle all its edges together and visually replace these edges with a rectangle to indicate an edge bundle.

3) *Connecting bundles with entities.* We link bundles and entities based on membership. This assures that entities and their associated bundles are fully connected (for $R1(c)$).

This bicluster-based edge bundling can potentially reduce visual clutter and clearly present a coordinated relationship (for $R1(b)$). As is shown in Figure 4.4, compared with (A), (B) clearly illustrates the coordinated relationship between four people and five phone numbers.

In fact, BiSet supports three modes to show edges: edge only mode ($EM$), hybrid mode ($HM$) and bundle only mode ($BM$), shown as (A), (B) and (C) respectively in Figure 4.4. In $EM$, BiSet shows edges without bundling. In $HM$, BiSet shows independent edges and bundles. In $BM$, BiSet just displays bundles. The three modes attempt to meet different user needs. For example, $EM$ potentially reveals the overview of relationships between two entity sets (e.g., (A) in Figure 4.4). $BM$ enables users to focus on analysis just with coordinated relationships. $HM$ can help to visually organize groups of individual relations into multiple
Figure 4.4: Three modes in the “in-between” layer for displaying edges. (A) is the edge only mode that shows all edges between related entities. (B) is the hybrid mode, which presents bundles with individual edges. (C) is the bundle only mode that just displays bundles.

levels (e.g., coordinated bundles with individual entity-level relationships). In BiSet, users can switch modes during their analysis. An example of using semantic edge bundling in BiSet is shown in Figure 4.5, which reduces 164 edges to 9 bundles. In this example, we use LCM to calculate biclusters and set the minimum support parameter to three, which assures that each calculated bicluster has at least three entities in one of the two related domains (here is the people’s name).

In addition to improving readability, bundles in BiSet preserve the coordinated attribute
Figure 4.5: A semantic edge bundling example in BiSet. (A) shows the original 164 edges. (B) After semantic edge bundling, there are 9 bundles. Of entities and edges. This enables users to infer semantic meanings about potentially coordinated activities. For example, why are the four people all related with the five phone numbers in Figure 4.4? Perhaps they colluded about a terrorist assault. Such semantic insights cannot be easily revealed from separated entity-level or group-level of relations. Thus, edge bundles in BiSet serve two important roles: improving readability and revealing semantic insights.
Visual Encoding in BiSet

BiSet uses four major visual channels [91] to encode bundles, entities and edges: shape, size, color and position. Figure 4.6 shows a detailed example of visual encodings in BiSet.

Figure 4.6: Visual encodings in BiSet. (1), (2) and (3) present the normal state of an entity, a bundle and edges, respectively. (4) and (4’) show the selected state of an entity and a bundle with accumulated highlighting. (5) and (5’) present the mouseover state of an entity and a bundle. (6) shows accumulated highlighting of an entity. (7) presents the highlighting state of edges. (8) shows the accumulated highlighting of edges.

Shape and Size In BiSet, entities and bundles are represented as rectangles (e.g., (1) and (2) in Figure 4.6). Edges are visualized as Bézier curves. We choose Bézier curves since they can generate more smooth edges, compared with polylines [83].

Length, width and font size are three specific types of size channel used in BiSet. Rectangles
indicating entities are equal in length, while those representing bundles are not. BiSet applies a linear mapping function to determine the length of a bundle based on the total number of its related entities. In a bundle, BiSet uses two colored ovals (light blue) to indicate the proportion between its related entities in the left list and those in the right list. In an entity rectangle, a small rectangle is presented on the left to indicate its frequency in a dataset. The length of these small rectangles is determined by the frequency of the associated entities. Based on these with different color encoding and position, users can easily discriminate entities from bundles (for \(R1(a)\)). In addition, the width of edges can reflect results of user selections. For instance, in Figure 4.6, compared with the width of those in (3), the width of edges in (8) is larger, since two relevant entities are selected. Moreover, when hovering an selected entity or bundle, related entities will be displayed in larger fonts. This helps users to review relevant information of previous selections (for \(R1(c)\)).

**Color Coding** BiSet applies color coding to entities, bundles and edges to indicate their states. In BiSet, entities and bundles have three different states (normal, mouseover and selected), and edges has two different states (normal and highlighting). In Figure 4.6, (1), (2) and (3) respectively present the normal state of an entity, a bundle and edges; (4) and (4’) show the selected state of an entity and a bundle; (5) and (5’) illustrate the mouseover state of an entity and a bundle; and (7) and (8) demonstrate the highlighting state of edges. In addition, two different colored borders (blue and black) are used to help users further discriminate the mouseover state from the selected state (for \(R1(d)\)). When hovering an
entity or a bundle, a blue border will be added to the rectangle. This border will change to black after user selection, which indicates that the state has changed from mouseover to selected.

*Accumulated highlighting* is important in BiSet, which is triggered by mouseover and selection. Different from simple highlighting, accumulated highlighting provides useful visual clues (e.g., darker in orange) for shared entities (for $R1(c)$) and bundles. BiSet applies accumulated highlighting to entities, bundles and edges by increasing the shading of their colors. For example, in Figure 4.6, the entity in (6) is in darker orange, compared with those in (4) and (5), since its highlighting is accumulated based on selections of the entity in (4) and AMTRAK, and the mouseover on (5).

**Position** Position is used to organize entities and bundles in BiSet. A set of entities of a certain domain (e.g., people) is organized as an entity-list. In between two neighboring entity-lists (the “in-between” layer), there is one relationship-list that contains coordinated relationships as biclusters (visually as edge bundles).

In entity-lists, the positions of entities can be determined in three ways (for $R3$): in an *alphabetical order*, based on frequency, or based on the order of (one-side) associated bundles. Alphabetical and frequency-based ordering can help to organize entities. However, they may lead to a severe problem of membership separation, since entities belonging to the same bicluster may not be listed close to each other. This results from the trade-off discussed in Section 4.1.1. To balance this trade-off, BiSet provides the third approach to organize the
position of entities based on the order of (one-side) associated bundles. For example, entities in the left list in Figure 4.6 are ordered based on associated biclusters in the middle list.

The larger a bicluster’s size is, the more important it is likely to be. A bicluster in larger size contains more information, so it is more likely to reveal potentially meaningful coordinated relationships. With this rationale, we apply a greedy algorithm, listed below, to organize entities based on the size of their associated biclusters.

BiSet also allows automatically changing positions of groups of entities (for R3) by dragging a bundle in the “in-between” layer. Figure 4.7 shows such an example. After dragging a bundle, two groups of entities in its two neighboring lists automatically move to new positions.

In the “in-between” layer, BiSet supports two ways to organize positions of bundles: automatically adjusting positions based on related entities, and manually dragging and moving bundles to new positions. The former allows listing bundles based on the positions of entities in either or both neighboring list(s), while the latter enables users to adjust the automatically generated layout based on their ad hoc needs (e.g., synthesis with created spatializations).

4.2.3 Interaction: Exploring and Organizing Information

*Revealing and organizing information in a bidirectional manner* serves as a key design principle for interactions in BiSet. By enabling users to directly interact with entities in entity-lists, and bundles in the “in-between” layer, BiSet can potentially support four types of exploration
Algorithm 1: Get positions of entities associated with biclusters

**input**: `curPos`, initialized as 0 for the top position

- `orderedBics`, a list of ordered biclusters
- `bicEntDict`, a dictionary stores entities for each bicluster
- `entBicDict`, a dictionary stores biclusters for each entity
- `rankedEntSet`, a set stores entities already ranked

```plaintext
foreach bic in orderedBics do
    entList = bicEntDict(bic);
    foreach entity in entList do
        if not entity in rankedEntSet then
            bicList = entBicDict(entity);
            totalRank = 0, num = 0;
            foreach bicluster in bicList do
                totalRank += bicluster.rank;
                num += 1;
            end
            entity.rank = totalRank / num;
            rankedEntSet.add(entity);
        end
    end
end

orderedEnts = entList.sort(self.rank);

foreach entity in orderedEnts do
    entity.pos = curPos++
end
```
Figure 4.7: Dragging a bundle in the “in-between” layer. Entities associated with this bundle automatically move to their new positions.

and two ways of organizing information.

BiSet supports bidirectional information revealing. Specifically, users can find relevant bundles by selecting or hovering over entities, and vice versa (for R2, ii and iv). Relevant entities or bundles in entity-lists or “in-between” layer will also be highlighted in BiSet (for R2, i and iii), when users interact with an entity or a bundle. This means that when users select (or hover over) entities, BiSet can use accumulated highlighting to reveal four types...
of potentially relevant information: 1) entities in the same list that belong to the same bicluster(s) with the selected entities, 2) entities in other list(s) that are related with the selected entities, 3) bundles in neighboring “in-between” layer that are directly connected with the selected entities, and 4) bundles in other relationship-list(s) that are associated with the selected entities via bicluster-chain(s). 1) and 2) satisfy the requirement of T1 and T2 (from entity to entity), and 3) and 4) support T3 and T4 (from entity to relationship).

Similarly, when users select (or hover over) bundles, BiSet can reveal the same four types of information, but they are related with selected bundles. In this case, 1) and 2) fulfill the requirement of T7 and T8 (from relationship to entity), and 3) and 4) can support T5 and T6 (from relationship to relationship). For example, in Figure 4.6 when users hover the entity in (5), three entities in the left list and six entities in the right list are highlighted. Of the six entities, the FBI one is in darker orange, which indicates that it is shared with another bundle (bicluster) on the bottom. The bundle on top is also highlighted, since it is directly related to the entity. With such ways of revealing information, BiSet can support the four types of exploration.

**Organizing Information**  BiSet supports two ways of organizing information, which is also bidirectional. Users can organize the position of entities based on bundles, and vice versa. BiSet uses vertical positions to visually externalize the organized entities and bundles. As discussed above, for bundles in the “in-between” layer, BiSet can not only automatically organize them based on the position of related entities, but also enable users to manually
adjust their positions by dragging and moving them. For entities, BiSet provides three options to automatically order them in an entity-list. When users drag and move a bundle, associated entities in two neighboring lists move with it, as is shown in Figure 4.7. This enables users to manually adjust positions of a group of related entities by using bundles. Thus, in BiSet, entities and bundles can mutually impact each other, which provides a flexible way for users to organize information in lists.

Figure 4.8: The document view from bundles in BiSet. (A) shows bicluster ID, related document and associated entities. (B) shows the content of a document. (C) lists all document ID(s) in the data with a search function.

BiSet also allows users to review documents directly from bundles and entities with a right click menu (for R4), shown in Figure 4.8. When finding an interesting bundle or entity, users can use a right click menu to open a popup view where relevant documents are listed. This view is on top of the view for relationship exploration. After reading, users can quickly return to previous view by closing it.
Chapter 5

Advanced Interactions to Scale Up

The BiSet [123] technique, discussed in the previous chapter, provides a clear way to visually present both entities and relationships. However, as the number of biclusters increases, the list of biclusters grows. When overlaps among biclusters get complex, there are more edges connecting biclusters with related entities. These may lead to a long list of biclusters with numerous edge crossings. In this case, users may have to manually move bundles to reduce edge crossings and scroll their views back and force to navigate in this long list of biclusters. To find similar biclusters, users need manually check shared entities between biclusters. Moreover, BiSet allows users to see related biclusters with connection based highlighting, as they select a bicluster. However, many biclusters may be highlighted, when they select one from a long bicluster list with many overlaps. Due to lacking functions to prioritize these highlighted biclusters and direct users towards promising ones, in BiSet, users have to repetitiously check all related biclusters based on shared entities. Thus, BiSet still needs
significant amount of effort from users to explore coordinated relationships.

This leads to four key challenges for bicluster visualizations to support coordinated relationships exploration, when facing the above situation: 1) reducing visual clutter, 2) decreasing the number of biclusters on screen, 3) prioritizing biclusters or bicluster-chains, and 4) revealing similarity among biclusters. Specifically, to serve user exploration, how can we organize biclusters and entities to reduce edge crossings? How can we reduce the number of biclusters on screen to ease navigation? How can we prioritize biclusters or bicluster-chains to guide user exploration? How can we show similarity among biclusters to users?

To address these challenges of using biclusters to support exploring coordinated relationships, in this chapter, we present four ways of interacting with biclusters: seriation, aggregation, prioritization and attraction.

5.1 Requirement Description

We identify the following four requirements, which correspond to the four questions mentioned above. These questions reflect key concerns raised as the number of biclusters increases and the complexity of their overlaps grows.

**R1: Organizing elements in multiple lists.** How can we organize biclusters and entities to reduce edge crossings in lists based layout? In BiSet, the position of biclusters and their associated entities can impact edge crossings, because bicluster overlaps are revealed by edges from the same entities connecting with different biclusters. To reduce edge crossings, in the
most simple case, we need to simultaneously organize elements in three lists: a bicluster-list and its two neighboring entity-lists. This is challenging for two reasons. It requires sorting in a 3D space, if we consider each list as an individual dimension. Moreover, positions of elements in each dimension are constrained by positions of elements in other dimensions, if we want to put biclusters and their associated entities close to each other. BiSet visually displays the three lists in a linear manner: a bicluster-list in-between two entity-lists. Thus, this problem can be viewed as sorting two pairs of neighboring lists: a bicluster-list with its left neighboring entity-list, and a bicluster-list with its right neighboring entity-list.

For each pair of lists, minimizing the number of edge crossings by ordering elements in two lists is NP-hard [37], which pursues heuristic solutions. Seriation is a possible solution [115]. As discussed before, seriation has been used to reveal patterns in a matrix. With some heuristic strategies to permute rows and columns in a matrix, seriation attempts to order elements in rows and those in columns in two sequences and the combination of the two can reveal some patterns in the matrix [87]. However, the above problem cannot be solved by simply applying seriation to the two different pairs of lists, because biclusters in the bicluster-list may have two different orders. One comes from the seriation between the bicluster-list and its left neighboring entity-list, and the other is from the seriation between the bicluster-list and its right neighboring entity-list. Because of two different orders, how to organize biclusters in this bicluster-list becomes a problem.

R2: Reducing the number of biclusters to ease navigation. How can we reduce the number of biclusters on screen to ease user navigation? As the number of bicluster increases,
in BiSet, the bicluster-lists grows. In this case, users may have to scroll their views back and forth to navigate in a list. Reducing the number of biclusters on screen can shorten lists, which requires less navigation effort [47]. However, it is not reasonable to randomly remove some biclusters, since it may cause losing some potentially useful information. Thus, besides reducing the number of biclusters, we need to preserve and reveal the relationships in those biclusters that are removed from the screen.

**R3: Ranking biclusters or chains for guiding user exploration.** How can we prioritize biclusters or chains to guide user exploration? Similar to Jigsaw’s problem in finding biclusters (e.g., which entities to click for finding biclusters), BiSet does not support prioritizing biclusters or chains to guide user exploration, so users have to manually check low level information (e.g., related entities) to identify useful ones. Compared with Jigsaw, BiSet allows users to see algorithmically mined biclusters, which saves them from manually finding biclusters. However, compared with exploring entity-level relations for bicluster discovery in Jigsaw, manually checking bicluster-level information in BiSet, without much support, may take more cognitive effort. Thus, prioritizing biclusters and chains is needed. Previous study finds that users tend to use high frequency entities in biclusters for sensemaking tasks [121]. This indicates that rankings can impact the information that users choose to use. Based on this, ranking biclusters or chains is a feasible way to prioritize them to guide user exploration. Moreover, users progressively get to learn more information during their analysis process. To guide user exploration, the ranking of biclusters or chains should be adapted based on the information that users have learnt.
R4: Revealing similarity among biclusters. How can we show similar biclusters to users and reveal similarity difference among biclusters? Finding relevant information is an identified analytics strategy, which is commonly used in sensemaking tasks [79]. It can assist user analysis by broadening the set of information that has been learnt (e.g., finding additional people that are related to an identified suspicious group). Considering the overlaps among biclusters, specifically in BiSet, finding similar biclusters in the same bicluster-list may support such information broadening oriented exploration. Thus, based on a user specified bicluster, finding its similar ones is needed. In addition, users also need support for helping them to differentiate these similar biclusters (e.g., which one of the similar biclusters is the most or least similar to the user specified one). Such differences may help users to decide which similar ones to use for further analysis.

5.2 Four Ways of Interacting with Biclusters

To satisfy the above requirements, we present four ways of interacting with biclusters: seriation, aggregation, prioritization and attraction.

5.2.1 Seriation in Lists

Inspired by the design of data context map [24], we merge adjacency matrices, which represent relationships between entities and biclusters in neighboring lists, to enable seriation in lists based layout (for R1). By fusing four derived distance matrices (containing the pairwise
distance of data items, attributes, attributes to data items and data items to attributes) from a data matrix, data context map can display both data items and attributes in a 2D space. Using merged adjacency matrices, the double orders problem, discussed in Section 5.1, can be addressed, so it is possible to apply seriation to lists. Specifically, we use Correspondence Analysis (henceforth, CA) for seriation.

**Correspondence Analysis and Edge Crossings**

In traditional application scenarios, CA is performed over the contingency table [1]. Categorical values represented by rows (or columns) of the contingency table is characterized by the frequency distributions of the corresponding rows (or columns), which is called profile in CA. CA identifies a low dimensional subspace of the entire profile space (e.g. a one-dimensional line or a two-dimensional plane), which maintains the majority of the dispersions of the original profiles [55]. If two profiles are close to each other in the original profile space, they would also be close to each other in the identified low dimensional subspace.

In the application of reducing edge crossings between paired entity-bicluster lists, a pair of lists can be represented as an adjacency matrix, which is binary. 1 indicates an entity is linked with a bicluster, while 0 means not. This matrix can be converted into a contingency table, and the profiles used in CA for entities and biclusters can be formulated. For two entities, if their connected biclusters are almost the same, their profiles in CA would be similar to each other. For two biclusters, if their associated entities are almost the same, their profiles in CA are also similar and they are close to each other in the profile space. By
performing CA on this adjacency matrix, similar entities are grouped together and close to each other on the axis identified by CA. Due to the symmetric property of CA [55], with respect to rows and columns of the contingency table, the corresponding biclusters on its identified axis by CA are also grouped and follow the similar order as that of entities. If we organize entities and biclusters in lists respectively based on their corresponding orders from CA, edge crossings between different groups of entities and biclusters will be reduced, compared with other random arrangement of entities and biclusters in the list.

**Key Steps to Enable Seriation in Lists**

This merged-matrices based seriation includes five key steps.

1) *Adjacency matrices preparation.* Based on relations in each pair of neighboring lists (an entity-list and a bicluster-list), we get an adjacency matrix, where rows are entity IDs in an entity-list, and columns are bicluster IDs in a bicluster-list. Each cell in such matrices has a value of 0 or 1, indicating whether an entity is linked with a bicluster. 1 means that they are connected and 0 means that they are not.

2) *Matrices fusion.* We merge these adjacency matrices to get a fused matrix, where rows are all entity IDs and columns are all bicluster IDs from all paired neighboring lists in the previous step. If an entity is not connected with a bicluster, we fill the corresponding cell with 0.

3) *Seriation on a fused matrix.* We apply CA to this merged matrix and get the seriated
orders of entities and biclusters (as global orders), respectively. Other seriation methods can also be used in this step. We choose CA for it can help to reduce edge crossings, as discussed above, and it has been studied for bipartite graph partitioning [143]. Moreover, based on implementations in [41], CA is effective and reasonably fast.

4) Local order generation. We get local orders of entities in entity-lists and local orders of biclusters in bicluster-lists based on the two global orders. For the two seriated sequences of entity IDs and bicluster IDs, we separate them into different entity-lists and bicluster-lists respectively, by their types. In each entity-list or bicluster-list, the order of entities or biclusters is determined based on their global orders.

5) Visual mapping. In each entity-list, entities are displayed based on their local orders in the list. In bicluster-lists, the position of biclusters is determined by the average position of their connected entities. This attempts to get symmetric layouts for biclusters and their associated entities for readability and aesthetic purposes [11].

This fused matrices based approach enables applying seriation to a list-based layout. If all pairs of neighboring entity-lists and bicluster-lists are included, this approach gives an organized layout for all entities and biclusters. If one bicluster-list and its two neighboring entity-lists are involved, this approach gets an organized layout for these biclusters and entities. For the same group of lists, the former potentially gets an organized layout for them from an overview perspective, while the latter gives them an organized layout from a focused perspective.
Figure 5.1: An example of organizing one bicluster-list and two entity-lists with two approaches. (A) gives the result of the greedy approach. (B) shows the result of the seriation approach.

The seriation approach, discussed above, can generate better visual layouts than the greedy approach, discussed in previous chapter. The greedy approach orders biclusters by size. Then, based on ordered biclusters, it assigns orders to related entities from the largest bicluster to the smallest one. If an entity is connected with multiple biclusters, its position is determined by the one with the largest size, which is a greedy approach. Finally, similar to the last step above, the position of biclusters is determined by the average position of their related entities. Figure 5.1 shows an example of organizing the same three lists with the two different approaches. In this example, there are 12 biclusters connected with 54 entities, and 9 of them have overlaps with other biclusters by shared entities. In total, this leads to 71
edges that connect entities with these biclusters. In Figure 5.1, (A) shows the result from the greedy approach used in BiSet, while (B) gives the result from the seriation approach. Visually, the layout in (B) is better than that in (A). In fact, the number of edge crossings in (A) is 169, while in (B), the number is 45, which is significantly reduced (by around 73%). In addition, since the position of entities are better organized in (B) than that in (A), the seriation approach potentially helps to address the visual overlapping problem of biclusters (e.g., two groups of biclusters overlapping with each other in (A)). Thus, compared with the greedy approach, this fused-matrices based seriation can help to reduce edge crossings and better organize entities and biclusters in lists.

5.2.2 Aggregation

Aggregation helps to reduce the number of biclusters on screen and preserve the information in removed ones (for $R2$). It enables merging similar biclusters based on Jaccard index. We use the equation (2) in Section 2.3 to control bicluster aggregation. To support aggregation, for each bicluster-list, we provide two sliders: a weight slider and a threshold slider. The former enables users to control the weight used in the equation for computing Jaccard index. For instance, the more users slide it to the right, the more emphasis is given to the similarity between sets of entities on the right (e.g., entities in the right neighboring entity-list). The latter allows users to control the threshold for aggregation. For example, if the computed Jaccard index between two biclusters is greater than the selected value in the threshold slider, we merge them. Moreover, the range of both sliders is from 0 to 1 (default setting is 0.5).
Based on values selected in the two sliders, we enable merging biclusters both *automatically* and *manually*, and they share the same visual encodings of merged biclusters.

Figure 5.2: An example of detailed visual encodings. (A), (B) and (C) present a bicluster, a merged bicluster, a merged bicluster (under selection), respectively. (D) and (E) show edges connecting merged biclusters with related entities, and those in (E) are highlighted. (F) and (G) show entities in the normal state and those in connection based highlighting state.

An example of detailed visual encodings to support bicluster aggregation is shown in Figure 5.2. In this example, there are one bicluster-list and two neighboring entity-lists. For entities, we keep the same visual encodings as those used in BiSet. In Figure 5.2, each entity is displayed as a blue rectangle and a small rectangle on the left of an entity reveals its frequency in a dataset (see (F)). Biclusters are represented as two horizontally adjacent rectangles with rounded corners. The length of these rectangles indicates the number of related entities (on the left or right), and the height of them are consistent for all normal
biclusters (e.g., (A)). For a merged bicluster, the height of the two rectangles is determined by the number of normal biclusters aggregated. The more normal biclusters are merged, the higher these two rectangles are. For example, comparing two merged biclusters (B) and (C), (B) is generated by merging more normal biclusters than (C) is.

Based on lattice, discussed in Section 2.2, merged biclusters are not closed. When merging two biclusters, two entity sets of one bicluster (left and right) are added to the two corresponding sets of another one. Because two entity sets are increased together, for a merged bicluster, not every entity in one entity set (e.g., entities on the left) is related with all entities in another (e.g., entities on the right). To reveal this, a merged bicluster have two types of edges, connecting itself with its related entities, visually displayed as normal curves and dashed curves (see (D)). The former indicates that all normal biclusters, involved in a merging, share an entity (e.g., 20 April, 2003), while the latter reveals that not all normal biclusters, involved in a merging, are related to an entity. The width of dashed curves is determined by the number of normal biclusters that share an entity. For example, for the merged bicluster (B), the number of its normal biclusters related with B. Dhaliwal, is larger than that sharing S. Albakri. Moreover, as users select a merged bicluster, its related entities are highlighted. In consistent with dashed curves, entities, shared by more normal biclusters involved in an aggregation, are highlighted with more bright colors (see (G)). With these visual encodings, a merged bicluster with associated edges can reveal the information of those normal biclusters that are aggregated.

Based on selected values using the two sliders, as discussed before, we enable two ways of
Algorithm 2: Similarity based aggregation in a bicluster-list

**input**: bics, a set of biclusters

- \( jMatrix \), a matrix of pairwise Jaccard index of biclusters
- \( \text{thresVal} \), a threshold to control bicluster aggregation

**output**: bicSets, a set of different sets of biclusters to be merged

```java
while bics is not empty do
    aSet.add(bics.pop());
    foreach bic in bics do
        movingFlag = 0;
        foreach member in aSet do
            if \( jMatrix[member][bic] \geq \text{thresVal} \) then
                movingFlag += 1;
            end
        end
        if movingFlag = aSet.length then
            aSet.add(bic);
        end
    end
    bicSets.add(aSet);
    while aSet is not empty do
        bics.remove(aSet.pop());
    end
end
```
merging biclusters: automatic aggregation and manual aggregation. The former takes a greedy approach, with its complexity of $O(n)$, to automatically merge biclusters, as listed above, which partitions a bicluster-list into groups of biclusters for aggregation. The latter allows users to specify biclusters for merging by using right click menus or with the attraction interaction, discussed in Section 5.2.4. After aggregation, if needed, users can split a merged bicluster using a right click menu on it, which shows normal biclusters that are previously merged. An example of automatic aggregation is shown in Figure 5.3, with both weight and threshold selected as 0.1. Compared with (A), (B) has less number of biclusters with fewer edge crossings, since similar biclusters are automatically merged. By splitting merged biclusters in (B), users can get corresponding groups of similar biclusters.

![Figure 5.3: An example of automatic aggregation. (A) shows the layout after seriation. (B) is the result of applying aggregation to the layout in (A).](image)
5.2.3 Prioritization

Prioritization potentially helps to direct users toward promising biclusters or chains that can lead further analysis (for R3). We use the MaxEnt based evaluation [138] to support prioritizing biclusters or chains, by ranking them based on surprisingness, discussed in Section 2.7.

In this MaxEnt principle based evaluation, biclusters and chains are presented as tiles [48], where a tile on a binary relationship is a sub-matrix in a data matrix with its cell values of 0 or 1. This evaluation approach dynamically maintains a MaxEnt model that is inferred from known biclusters or chains. This model represents the prior information of data. At the initial stage, when nothing is learnt, it is a uniform distribution. For (unknown) biclusters or chains requested for evaluation, the Kullback-Leibler (KL) divergence [14] between two stages (current and previous) of this MaxEnt model is computed. In the current stage, the MaxEnt model is a MaxEnt distribution inferred from the information including both known biclusters and chains and those currently under evaluation. In the previous stage, the MaxEnt model is a MaxEnt distribution inferred just based on the information of known biclusters and chains. The computed KL divergence gives evaluation results. A larger KL divergence indicates that a bicluster or chain is more surprising based on known ones, which means that this bicluster or chain can bring more new and related (to some extent) information. Moreover, if evaluated biclusters or chains are marked as known, the maintained MaxEnt model will be updated, by incorporating their information, and used for next evaluation. For the sake of space and the scope of this paper, please refer to [138] for more detail (e.g., entropy calculation and tile based MaxEnt distribution calculation).
We use this evaluation based on two key considerations: incremental knowledge building and expanding working memory. As discussed above, in this evaluation approach, a Max-Ent model that represents the learnt knowledge of data is maintained and can be dynamically updated, so it potentially enables users to iteratively interact with computations to drive evaluation. In each iteration, this MaxEnt model is updated by incorporating the information of biclusters or chains that users confirm as useful. Here, useful indicates that users have learnt such information. In this user-driven, iterative process, this MaxEnt model incrementally incorporates new pieces of information in useful biclusters or chains, so this model progressively stores the analytics provenance [94] of users (e.g., the information that users confirm as useful). In addition, after each iteration, the updated MaxEnt model is used for further evaluation. Since this updated model has included all useful information identified in previous investigations, it potentially expands the working memory of users. Compared with a manual evaluation, this MaxEnt model can consider all the learnt information to give more reasonable results (e.g., it is difficult for users to remember all the information from 10 or more useful biclusters and use that for further bicluster evaluation).

To enable user-driven, MaxEnt based evaluations for bicluster prioritization, we allow users to explicitly request evaluating related biclusters (e.g., those with shared entities) and chains (e.g., those passing through a bicluster) from a bicluster. Evaluations at two levels are enabled: bicluster-level and chain-level. From a bicluster, the former evaluates its related ones in the same bicluster-list and in its neighboring bicluster-lists, while the latter evaluates all possible chains that go through this bicluster. Users can request these evaluations from a
normal or merged bicluster using a right click menu. Moreover, additional visual encoding, surprisingness based highlighting, is added to BiSet to reveal evaluation results. Figure 5.4 presents an example of visual encodings that support prioritization (requested from a normal bicluster). Besides the default state (e.g., (D)), each bicluster or entity has two highlighting states: connection based highlighting and surprisingness based highlighting. For example, entities in (F) are in the connection based highlighting state since their connected bicluster (A) is selected, entities in (E) are in the surprisingness based highlighting state, because they are in the most surprising chain from (A), which is computationally identified.

Figure 5.4: An example of visual encodings used in prioritization. (A) is a bicluster that users request for the two levels of evaluations. (B) and (B’) respectively shows the most surprising biclusters in the same list and in a neighboring list. (C) and (C’) presents the least surprising biclusters in the same list and in a neighboring list. (D) shows normal biclusters that are not highlighted. (E) and (F), respectively, show surprisingness based and connection based entity highlighting. (G) presents the links in the most surprising chain.

There are three key steps in the bicluster-level evaluation: 1) related bicluster search, 2)
related bicluster evaluation, and 3) related bicluster coloring. For a user specified bicluster, its related biclusters are found base on the equation (1) in Section 2.3. If the Jaccard index between a bicluster and the user specified one is above 0, we consider it a related one. We use this loose constrain to assure that each bicluster that shares entities with the user specified one is considered for evaluation, which avoids missing surprising biclusters that share few entities with the user specified one. For each related bicluster, we derive its corresponding tile, and use it as the input of a MaxEnt based evaluation. Based on evaluation results, we rank related biclusters in each involved bicluster-list respectively, and use a linear mapping function to reveal their rankings with colors. The more surprising a bicluster is, the darker purple it is. For example, in Figure 5.4, (A) is a bicluster requested for a bicluster-level evaluation of its related biclusters. Based on this evaluation, (B) and (C) are the most and the least surprising one in the same bicluster-list respectively, while in a neighboring list, (B') and (C') are respectively the most and the least surprising one.

The chain-level evaluation also has three steps: 1) chain search, 2) chain evaluation, and 3) chain rank. In the chain search step, all possible chains passing through a user specified bicluster are found. Similar to a tree search, the user requested bicluster is treated as a root node. Then, a depth-first searching is conducted to find all chains starting from this bicluster. If the user specified bicluster for evaluation is not in the left or right most bicluster-list, we perform bidirectional search and then combine identified chains in the left and those in the right together to get all chains that go through this bicluster. In the evaluation step, the identified chains are converted into a unique set of tiles and used as the input of the
MaxEnt based evaluation. Finally, in the chain rank step, these chains are ranked based on their evaluation results, and the top one is visually displayed, by highlighting its starting and ending biclusters, entities locating in the chain, and edges. For example, in Figure 5.4, the most surprising chain from bicluster (A) is revealed by its components (A), (E), (G) and (B’). Considering possible visual clutter and information overwhelming, we choose to highlight one chain. Compared with others, information in the most surprising chain can bring more new information to users, so it is more likely to impact user analysis. Thus, we highlight the most surprising one.

These two-level evaluations help to prioritize biclusters and chains from a bicluster that users request for evaluation. With surprisingness based highlighting to reveal computationally identified rankings, users can be directed towards potentially useful biclusters or chains. Users can use right click menus on biclusters to mark them as useful, and this updates the maintained MaxEnt model used for next evaluation.

5.2.4 Attraction

Attraction reveals similar biclusters to users and enables users to organize them and related entities based on an investigated one (for R4). We apply the dust and magnet visual metaphor [140] at two levels (bicluster-level and entity-level) to support attraction. When users drag a bicluster (as the magnet), similar biclusters with related entities (as the dust) automatically move with the dragged one. To be consistent with aggregation, the similarity
calculation is based on user selected values using the two sliders, discussed before. Moreover, using right clicking menus on biclusters, users can choose to enable or disable attraction (when they drag a bicluster).

Attraction, with manual aggregation, is illustrated in Figure 5.5. (A) and (B) show that similar biclusters move automatically as users drag a bicluster. This helps to reveal similar biclusters and visually separates similar biclusters from dissimilar ones (e.g., see the separated groups of biclusters in (B)). (C) presents that, after dragging, a similarity radar appears and entities related with this group of similar biclusters move to the position close to these biclusters (e.g., see the position of entities changes between (C) and (A)). The distance between each similar bicluster to the user dragged one is determined by the computed Jaccard index using a linear function. The larger Jaccard index a similar bicluster has, the shorter distance between it and the user dragged one. Moreover, if the size of a similar bicluster is larger (or smaller) than the user dragged one, it locates above (or below) the dragged one. Based on positions of similar biclusters, a similarity radar is displayed.

The similarity radar plays two major roles: 1) indicating similarity difference, and 2) supporting manual aggregation. A similarity radar is displayed as a group of concentric ring areas in red (with different brightness to reveal similarity difference), and the user dragged bicluster locates in its center. The number of ring areas is determined by the number of unique Jaccard index of these similar biclusters, which ensures that similar biclusters with different Jaccard index are associated with different ring areas in a similarity radar. The outer circle radius of a ring area is determined by the distance between the (farthest) similar
bicluster (if there are multiple) in it and the user dragged one. The more inner ring area (visually with more bright red) a similar bicluster locates, the more similar it is to the user dragged one. For example, in Figure 5.5 (C), the bicluster in the outmost ring area (below the center) is less similar to the user dragged bicluster than the three in other ring areas (above the center), and the size of the user dragged bicluster is smaller than the three above it but larger than the one below it. In a similarity radar, size difference can also be visually perceived (above or below the center), which attempts to help user analysis. A less similar bicluster in a small size is less likely to have enough related information to broaden analysis, so users may not take effort to explore it, or merge it into a group of other similar ones in bigger size.

The similarity radar enables users to merge a group of similar biclusters by clicking on ring areas. When users hover the mouse over a ring area, its boundary (a circle) is highlighted (see Figure 5.5 (D)). After users click on a ring area, biclusters within the highlighted boundary are merged (see (E)). These ring areas work as handles to support users manually merging a group of similar biclusters. Compared with using a right click menu on biclusters to merge one with another, this similarity radar based merging is more efficient. Users can merge a group of similar biclusters with one click. Moreover, by viewing the position of biclusters on the similarity radar for reference, compared with randomly merging one bicluster with another, this similarity radar based aggregation may lead to more reasonable merging decisions.
Figure 5.5: An example of attraction with manual aggregation. (A), (B) and (C) sequentially shows the visual attraction as users drag a bicluster. (D) and (E) shows manual bicluster aggregation with a similarity radar.

5.3 Usage Scenario

We walk through a text analytics scenario to illustrate how BiSet, with four advanced interactions, supports an analysts in discovering coordinated activities with bicluster focused analysis. In the scenario, we use The Sign of the Crescent dataset [68]. This dataset has 41 fictional intelligence reports regarding a coordinated terrorist plot in three cities, and 24 reports are relevant to these plots. Each plot involves at least four suspicious people. We use
LCM [129] to find biclusters from the dataset with the minimum support parameter set to 3. This assure that each bicluster has at least three entities from one domain. This leads to 337 biclusters (including 122 thin biclusters) from 284 unique entities and 495 relationships.

Suppose that Mark is an intelligence analyst. He is assigned a task to read intelligence reports and identify potential terrorist threats with key persons. He loads the data in BiSet, selects four identified domains (people, location, phone number and date) and starts his analysis. Figure 5.6 shows the key steps of Mark’s analytical process.

**Easy layout management with seriation and rapid discovery of a group of similar biclusters using automatic aggregation.** Mark begins analysis with exploring relationships between people and location. He feels that the initial layout (entities are listed in alphabetical order and biclusters are ordered by size) is not well organized due to many edge crossings, so he decides to reorganize them to get a better view with fewer edge crossings. With seriation, entities in the two lists and bicluster in-between them are reordered. After that, he can clearly see that entities of the same biclusters almost locate near each other. However, Mark feels that the number of biclusters in-between the two lists is large and wants to merge them to get a better view with fewer biclusters. He adjusts the threshold slider to the left (reducing the threshold) and check his view. As he reduces the threshold value to 0.256, a big merged bicluster appears, shown in Figure 5.6 (1). This almost instantly catches his attention, so Mark stops changing the slider and starts to investigate this merged one. One question that he has is what this merged bicluster consists of. Using a right click menu, Mark splits this merged bicluster. Four similar biclusters appear, shown in Figure 5.6 (2),
and two of them share a high-frequency entity, A. Ramazi. High frequency indicates that this person is mentioned multiple times in the reports. Thus, Mark decides to investigate these two biclusters to see what they can further lead to.

**Effective way leading with prioritization.** Looking through edges from the two biclusters, Mark realizes that they are linked with many entities, so it is not an efficient strategy for him to check these entities. He decides to find what other biclusters they can lead to. Using a right click menu, Mark requests to evaluate their related biclusters, respectively. After each request, BiSet highlights their related entities based on evaluation results. One interesting thing that catches his attention is that the most surprising bicluster, computationally identified based on the two biclusters, are the same (Figure 5.6 (3) and (4)), and it does not share many entities with the group of similar biclusters found previously. Mark decides to follow this leads and investigate what additional information it is related with. By hovering the mouse over this bicluster, with connection based highlighting, he finds that it connects with four biclusters in the neighboring bicluster-list (between location and phone number) and two more in the further bicluster-list (between phone number and date). In total, this involves eight chains. To find which one to start with, Mark requests to perform a chain-level evaluation from this surprising bicluster. BiSet highlights the chain that identified as most surprising. The chain includes information from the four lists (Figure 5.6 (5)). Mark thinks that such information is possible to lead a story, so he decides to read the intelligence reports associated with biclusters in this chain. Using the right click menu on biclusters, BiSet allows to show related reports from biclusters in its document view [123].
The three bundles direct Mark to nine reports in total, and eight of them are relevant to each other. Referring to the eight reports, Mark identifies a potential threat plot with four key persons as follows:

\textit{F. Goba, M. Galab and Y. Mosed}, following the commands from \textit{A. Ramazi}, plan to attack \textit{AMTRAK Train 19} at 9:00 am on April 30.

\textbf{Branching analysis.} Mark is satisfied with this finding and marks the three biclusters in this surprising chain as useful. This informs the integrated MaxEnt model in BiSet that information in these biclusters has been learnt, and the model updates its background information for further evaluations. One report, from the bicluster in the middle of the surprising chain (the one with a black box in Figure 5.6 (5)), is irrelevant to that of the other eight, but the entities extracted from this report are connected with those in the identified threat. Thus, Mark considers the information in this report as potentially useful clues. To explore what these clues can lead to, he requests bicluster-level evaluation based on the bicluster in the middle of the surprising chain (Figure 5.6 (5)). After that, a bicluster, different from previously explored, is highlighted with darkest purple (the one with black box in Figure 5.6 (6)). By hovering mouse over this bicluster, different group of people and locations, and another bicluster are highlighted (Figure 5.6 (7)). After seeing this, Mark realizes that this may lead to some different plot(s).

\textbf{Conveniently revealing similar biclusters with attraction and flexibly merging biclusters with a similarity radar.} The connection based highlighted bicluster in (Figure 5.6 (7)) is in the group of similar biclusters found previously (Figure 5.6 (2)). Mark feel
that this is promising. He decides to move on with the highlighted one in (7) and explore its similar biclusters. Using a right click menu, He enables attraction on it. As he drags it, its similar biclusters follow. After he drops it, a similarity radar appears (Figure 5.6 (8)). By looking at the radar, Mark quickly finds the most similar bicluster to the dragged one. Moreover, he finds that they are different from each other by just one people and one location, so Mark thinks that they are quite similar and decides to merge them. By clicking on the inner circle of the radar, the two biclusters are merged (Figure 5.6 (9)).

This manually merged bicluster is connected with many biclusters in its neighboring bicluster-list. With the same strategy, Mark requests a chain-level evaluation from this merged bicluster. By looking through the newly identified chain, he finds that this chain is merged with previously identified one (Figure 5.6 (5)). Considering its possibility of leading to some related plot(s), Mark decides to check related intelligence reports from biclusters in this chain and find why they are merged. From these biclusters, Mark finds 10 unique reports in total. 6 of them show evidence of a new threat and three are relevant to previously identified threat plot. Based on these reports, Mark identifies another potential threat as follows:

_B. Dhaliwal and A. Ramazi_ plan to attack the New York Stock Exchange at 9:00 am on April 30.

Considering connections between this plot and the previously identified one (e.g., shared people and date), Mark also confirms that _A. Ramazi_ is the key person who coordinates the two planned attacks.
Figure 5.6: Key steps in finding two coordinated threat plots. (1): Using seriation to organize biclusters and entities and merging biclusters. (2): Splitting a merged bicluster, 4 normal biclusters appear and two of them are linked with A. Ramazi. (3) and (4): Evaluating related biclusters of the two biclusters noticed in (2) and their top ranked ones are the same. (5): From this shared top ranked bicluster, the most surprising chain leads to one threat plot. (6): From another bicluster in the most surprising chain identified in (5), its related biclusters in a neighboring bicluster-list are ranked. (7): The top ranked bicluster in (6) leads to another connected bicluster. (8): Similar biclusters of the bicluster noticed in (7) are shown in a similarity radar and one of them is connected with A. Ramazi. (9): From a merged bicluster, the most surprising chain leads to another threat plot.
Chapter 6

Evaluation

Biclusters help to reveal coordinated relationships in a dataset. Algorithms for bicluster identification often aim at finding closed biclusters. However, algorithmically mined biclusters are not exclusive with each other. Because of such overlaps, exploring multiple overlapped closed-biclusters and merging useful information from them can help analysts to make analytical hypotheses, beyond simply getting to know fact of data. For example, analysts may aggregate two biclusters, they each indicating a set of people working at a set of IT companies. This aggregation reflects an inferred insight of people with similar working experience. Moreover, such analytical tasks potentially lead to partial biclusters exploration, because in a user generated bicluster, not every people is connected with all IT companies.

While computation (e.g., biclustering) can find biclusters from a dataset, it is not trivial to enable selectively aggregating useful information from multiple biclusters in an automatic
way. Human effort (e.g., domain knowledge) is a necessary input to support this type of sensemaking tasks. BiSet provides a clear way to visually display both entities and coordinated relationships (as closed biclusters), where biclusters are presented as edge bundles. Because entities and biclusters are displayed in different lists, users can easily distinguish them. However, how this technique impacts user exploration of coordinated relationships (including partial biclusters) is still unknown. To study this and further inform the design of future visual analytics tool to support exploring coordinated relationships, we conducted a user study.

The purpose of this study is to investigate the effectiveness of two bicluster-based techniques of BiSet, specifically edge bundling and seriated ordering, in supporting coordinated relationships exploration. In the study, we compare BiSet with a traditional list visualization, particularly similar to the list view in Jigsaw [117]. We focus on two key research questions, listed as follows:

- Does BiSet with the two specific techniques, compared with traditional list visualizations without them, help users more efficiently explore coordinated relations?

- Comparing BiSet (with the two specific techniques) with a traditional list view (without them), what are trade-offs when using them for coordinated relationship exploration?
6.1 Participants and apparatus

We recruited 20 graduate students (9 males and 11 females) from a university, aged 24-33 (mean 28). Participants were from various departments, such as business management, civil engineering, computer science, food science and psychology. None of them had prior experience with biclusters. All participants had normal or corrected-to-normal vision without color vision deficiency. The experiments were performed on a laptop, specifically, a Macbook Pro with 2.3 GHz Intel Core i7 processor and 16GB memory. The visualization was displayed on the 15.4-inch laptop screen with Google Chrome, version 51.0.2704.106 (64-bit). The visualization fitted in the screen, so no scrolling interaction was needed. All participants completed study tasks with a mouse and a keyboard.

6.2 Data

We use synthetic data in this study to ensure generalizability of the results. In total, we generate four datasets, for the four experiment conditions in the study. These datasets have the same level of complexity based on graph connectivity, and each of them contains two sets of entities with different types (e.g., person and location). The size of the datasets and complexity of their connectivity is determined by considering results of a pilot study, discussed as follows.
6.2.1 Dataset Preparation with a Pilot Study

In order to prepare datasets with a reasonable size and complexity for this study, we conducted a pilot study with four volunteers, who were all graduate students from a university. The settings of this pilot study (e.g., a laptop, a mouse and a keyboard) are the same as those used in the primary study. We generated two datasets for both views, with bundles and with bundles. Table 6.1 gives a brief summary of a dataset used in this pilot study. Each dataset has two lists of entities, and total number of entities in each list is 18. Entities in different datasets have different labels (e.g., people’s names). Two lists of entities in each dataset form 9 biclusters, and each entity belongs to at least one of these biclusters. As is shown in Table 6.1, these biclusters have three different sizes. We pick these sizes due to two reasons. First, biclusters in these sizes involve either 5 or 6 entities, which takes users a reasonable amount of time to understand. Second, these sizes can give a variety of overlap levels between two biclusters. These biclusters fall in three groups based on the number of shared entities by pairwise biclusters: 0, 2 and 4. In addition, biclusters in different groups do not overlap each other. In summary, each dataset has 36 entities and 53 individual relationships, and they are associated with at least one bicluster. We intentionally avoid isolated individual relations (e.g., individual relations do not belong to biclusters), since they can potentially increase cognitive effort of exploring coordinated relationships.

Similar to user tasks in the primary study, each participant was assigned two tasks, specifically, finding people with similar working experience, from the two generated datasets. The expected answers should contain at least three persons and at least three companies, as evi-
Table 6.1: A summary of a dataset used in the pilot study.

<table>
<thead>
<tr>
<th>Number of shared entities</th>
<th>Number of biclusters</th>
<th>Size of biclusters</th>
<th>Total entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>3<em>3, 3</em>2, 2*3</td>
<td>16 (8 + 8)</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3*2,</td>
<td>12 (6 + 6)</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2*3</td>
<td>8 (4 + 4)</td>
</tr>
</tbody>
</table>

dence of their hypotheses. Entities in the two datasets remain the same relative visual-order, although they are assigned with different labels (e.g., people’s names and company names).

The only difference controlled for the two tasks is the visual representation. One is visualized with edge bundles (condition 1, denoted as pilot-WB), while the other is displayed without edge bundles (condition 2, denoted as pilot-NB).

In this pilot study, we did not limit the time for participants to finish each task. Instead, we asked participants to find as many answers as they want, without giving them a specific number of expected answers. With this strategy, we wanted to explore when the participants would stop their analysis. After they finished a task, we reviewed their answers with them and asked them to justify their answers (e.g., explaining why they chose these people with those companies).

Results of this pilot study is shown in Table 6.2. On average, for both views, the four participants could find four answers, and the majority of these answers covers three closed biclusters. For the view without bundles, it took them more than 9 minutes to find four
answers, while using bundles, it took them almost half of the time to get the same amount of answers. Since most answers are closed biclusters, this indicates that participants tended to stop analysis after getting all three closed biclusters. This also helps to explain why they took less time to finish the task in pilot-WB than that in pilot-NB. Because bundles explicitly represent coordinated relationships, it is easier for participants to see closed biclusters.

<table>
<thead>
<tr>
<th></th>
<th>No bundle</th>
<th>With bundle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (min:sec)</td>
<td>5:36</td>
<td>1:43</td>
</tr>
<tr>
<td>Number of answer</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>12:25</td>
<td>11:27</td>
</tr>
<tr>
<td></td>
<td>11:24</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>7:48</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>9:19</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 6.2: A summary of the pilot study results.

Considering the time difference in task completion between the two views, the number of overlapped biclusters might be more complex for the view without bundle. It took participants almost double amount of time, on average, to finish the task in pilot-NB than that in pilot-WB. The more biclusters overlap each other, the more complex these related entities are (e.g., more involved entities with more edges). This leads to more user effort in exploring coordinated relationships in the view without edge bundles. In pilot-NB, users have to check
individual relationships and compare entity labels to find an answer, since there is not obvious visual clues for them to see it. This suggests that the number of overlapped biclusters in currently generated datasets might be overbalanced for the two views.

Based on results of this pilot study, we made changes in the primary study in two aspects: dataset generation and user tasks description. The former attempts to balance the complexity of datasets for both visual representations, while the latter aims at persuading users to keep working on analysis after getting all closed biclusters. The two datasets, used in the pilot study, lead to reasonable number of answers that can be meaningfully justified by the participants. We posit that participants would explore more if they could be directed with a more clear task description (e.g., giving them a specific number of expected answers). Moreover, with a longer period of exploration, more insightful results may be covered [93].

6.2.2 Dataset Generation for the Primary Study

For the primary study, we use four synthetic datasets. They have the same level of complexity, from the perspective of size and graph connectivity. Each dataset has two sets of entities. One entity-set is people’s name (visualized in the left list), and the other set is organization, location, item, or course (visualized in the right list). The combinations of the two sets result in four datasets. We generated datasets starting from biclusters.

Based on the pilot study results, we reduce the number of closed biclusters from 3 to 2, which attempts to balance the complexity for both views. This also potentially helps to
save user effort and time to find closed relationships. In addition, we increase the number of overlapped biclusters from 6 to 14 (including the two closed biclusters). For one thing, this increase the possibility for users to explore information from more than a single bicluster. For another, it better fits the time period (specifically, 20 minutes) for user tasks in the primary study.

Figure 6.1: Examples of three levels of overlaps based on shared entities.

Similar to that in the pilot study, besides the two closed ones (3*3), the size of biclusters is either 3*2 or 2*3. For each bicluster in the same dataset, its left entities are people’s names, and right entities are labels of organizations, locations, items or courses. For enabling semantic oriented information aggregation, in each dataset, we created 12 categories to
generate the right entity-set based on the 14 biclusters. 6 of these 14 biclusters, specifically two closed biclusters and four pairs of biclusters, are expected answers of users tasks in the primary study. Entities of these four pairs share the same one or two categories, regardless of their specific labels. Moreover, there are, in total, three levels of bicluster overlaps for the 14 biclusters: low, medium and high, which corresponds to sharing 1, 2 or 4 entity labels between two biclusters. Figure 6.1 shows examples of the three levels of overlaps.

In summary, for the primary study, we generated four datasets for the four experiment conditions. Each dataset has 58 entities with 83 individual connections in total, which leads to 14 biclusters. The size of these datasets (both entities and biclusters) is about 1.5 times of those used in the pilot study. This potentially fits the 20-minute time period for users to finish one task. The detailed information of these four datasets can be found in the appendix.

6.3 Methodology

We conducted a within-subject, 2×2 factorial study with four user tasks. The two key factors are visual representation and entity order. The former has two levels: with edge bundles and without edge bundles. The latter also contains two levels: random ordering and seriated ordering. Combinations of the two factors lead to four experiment conditions. For each condition, a user task is assigned using one of the four generated datasets. Considering the cost of time and effort, we do not replicate tasks for each experiment condition. The combination of factors corresponding to each task is summarized in Table 6.3. To avoid the
potential impact of task orders, sequence of these four user tasks are randomly shuffled.

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Visualization</th>
<th>Order</th>
<th>Condition Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>No bundle</td>
<td>Random</td>
<td>NR</td>
</tr>
<tr>
<td>T2</td>
<td>With bundle</td>
<td>Random</td>
<td>BR</td>
</tr>
<tr>
<td>T3</td>
<td>No bundle</td>
<td>Seriation</td>
<td>NS</td>
</tr>
<tr>
<td>T4</td>
<td>With bundle</td>
<td>Seriation</td>
<td>BS</td>
</tr>
</tbody>
</table>

Table 6.3: A summary of the factor combination for each user task.

The four user tasks are similar to each other, although different labels are used in different datasets. Specifically, descriptions of these tasks are listed as follows.

- T1: Grouping people with similar *working experience*.
- T2: Grouping people with similar *travel preference* based on their travel history.
- T3: Grouping people who have similar *shopping style* based on their shopping records.
- T4: Grouping people with similar *interest of learning* based on courses they have taken.

We require that each finding should contain a set of people’s names (from the left entity-list) and a set of entities from the right entity-list. The cardinality of both sets should be in the range from 3 to 6 (including the two boundaries). In addition, different from the pilot study, we informed participants that there were 6 expected answers for each task, but they were free to find as many as they could. These strategies attempted to avoid user stopping analysis after merely finding the two closed biclusters in each synthetic dataset.
We used the seriation approach, discussed before, to generate seriated orders for entities and biclusters (if displayed). For random ordering generation, we arbitrarily shuffled entities in the two entity-lists, and then determined the positions of biclusters by the average position of their associated entities. In total, we generated 100 samples of such random orders and randomly select one from them for both NR and NS. Thus, entities in NR and NS have the same relative orders based on random ordering, while entities in BR and BS remain the same visual sequences based on results of seriation.

6.3.1 Data Analysis Method

We took two data analysis strategies to gain results of this study: top-down and bottom-up. The former is applied for hypotheses test, and detailed descriptions of these hypotheses are discussed in Section 6.6. Specifically, we did two-way analysis of variance (ANOVA) for testing our hypotheses about the impact of the two factors in the study, visual representation and entity order. The latter follows the grounded theory [22] to identify potential trade-offs of the two compared techniques.

6.4 Procedure

The procedure of this study contains four parts. It began with a brief tutorial about coordinated relationships, biclusters and edge bundling. Then we used the two datasets of the pilot study to demonstrate the two visual representations and interactions. Second, as a training
session, participants were asked to find a 3*3 bicluster from two views using the two demo datasets. During this session, we helped them to resolve questions about the visualizations and interactions. Third, participants were informed the detailed task description and specific requirement for finding their answers, as discussed before. After that, they worked on the four tasks based on randomly assigned orders. For each task, participants had at most 20 minutes to find their answers, and a follow-up 10 minutes to review and justify their answers. Participants were allowed to have a short break after finishing each task, if needed. Thus, it took each participant about 2 hours to finish all four tasks. Finally, after they finished all the tasks, we interviewed with them to learn their analysis strategies, judgement of complexity of the four tasks, and the impact of edge bundles on their analysis.

6.5 Data Collection

Study data were collected from interaction log files, screen recording, observations and interviews. We logged four key types of user mouse interactions during their analysis: mouse-over an entity or a bicluster, mouse-out an entity or a bicluster, selecting or unselecting an entity or a bicluster, and adding or removing an entity to answers. For each interaction, four key components were logged: the time stamp, interaction type, the target object type (e.g., an entity or a bicluster), and the target object ID.
6.6 Hypotheses

We made the following seven hypotheses about user performance, considering the two visual representations and two entity orderings.

- **H1**: Under the same entity ordering condition, when using bundles, we expect that user findings are more likely to include closed biclusters and those with relatively higher level of overlaps. Since each bundle explicitly reveals a closed bicluster, it is easy for users to find closed biclusters that satisfies the task requirement. Moreover, bundles potentially help users to investigate overlaps between biclusters with edge groupings. Understanding such overlaps may lead to user selecting entities from overlapped biclusters as findings.

- **H2**: With the same entity ordering, comparing a bundle-enhanced list view with a traditional list view, we expect that users can find more justifiable answers, because bundles have already grouped some entities together.

- **H3**: We expect that more justifiable answers can be found with seriated ordering than random ordering, under the same representation condition. The former attempts to organize entities of the same bicluster(s) close to each other. With such computational results, it is more likely for users to find similar entities and group them together.

- **H4**: Under the same ordering condition, it takes users less entity visits to get justified answers with bundles than without bundles, since edge bundles reveal entity coalitions.
• **H5**: For the same visualization condition, compared with random ordering, we expect that it takes users less entity visits to identify an answer when using seriated orders. After seriation, similar entities (based on their associated biclusters) are visually listed close to each other, so it is easier for users to find similar ones.

• **H6**: With the same entity orders, we expect that participants take less time to get an justified answer using bundles than that without bundles.

• **H7**: Under the same representation condition, we expect that it takes users less time to find an answer when using seriated ordering than random ordering.

**H1** measures performance by variant findings (e.g., entities from closed biclusters or merged biclusters with different levels of overlaps), while **H2** and **H3** measure performance from the perspective of answer accuracy. Moreover, **H4**, **H5**, **H6** and **H7** consider performance from the perspective of user interaction effort and time.

### 6.7 Results

#### 6.7.1 Variance

Entities in each user finding potentially lead to two types of biclusters: a closed bicluster or a merged bicluster. Both of them indicate coordinated relationships identified by participants for the given tasks. Entities of a merged bicluster come from biclusters with three different
levels of overlaps: high, medium and low, as discussed before, which corresponds to the number of shared entities between two biclusters as 4, 2 and 1. Thus, there are four possible types of biclusters involved in user findings. A summary of the four types of biclusters in all submitted user answers is shown in Figure 6.2. The variance of them in user answers indicates user preference of finding coordinated relationships in different experiment conditions.

Figure 6.2: A summary of total number of found biclusters, corresponding to each of the four types, in all experiment conditions.

Bundles direct user attention to closed biclusters or those with relatively higher level of overlaps. We found significant effects of representation on biclusters involved in participant answers: closed biclusters ($F_{(1,76)} = 9.84$, $p = .0024 < .005$), merged biclusters with the low
overlap level \( (F_{(1,76)} = 5.64, \ p = .02 < .05) \), and merged biclusters with high and medium overlap levels \( (F_{(1,76)} = 8.66, \ p = .0037 < .005) \). Moreover, in Figure 6.2, under the same entity ordering condition, comparing NR with BR or NS with BS, using bundles leads to participants finding more closed biclusters and merged biclusters at the level of high and medium; while without bundles, participants found more merged biclusters with the low overlap level.

These results support \( H1 \). Bundles explicitly represent closed biclusters, so it is easier for participants to find closed ones with bundles. This explains the more total number of closed biclusters found when using bundles. For high level overlapped biclusters, when users select an entity, there are less number of entities highlighted, compared with biclusters with medium and low overlap levels. Due to less highlighted entities, it took participants less effort to investigate these entities. Thus, the total number of merged biclusters with the high overlap level do not vary much among the four conditions. For merged biclusters with the medium overlap level, we also found a significant impact of bundles \( (F_{(1,76)} = 7.4, \ p = .0081 < .01) \). Bundles enable users to see entity groupings, which helps them to check overlaps between groups. Without bundles, participants had to manually found entity groupings, which is not trivial. Due to lacking such grouping information, participants may simply select entities (e.g., from highlighted ones) that come from none-overlapped biclusters or those with a low overlap level. Thus, compared with using bundles, participant answers, in traditional list representations, include more merged biclusters with the low overlap level but less ones with the medium overlap level.
Considering these differences, participants may pay more attention to the overlap of biclusters when using bundles, because bundles potentially promote the awareness of computed entity groupings. Such awareness of groupings may lead participants to investigate overlaps between groups for finding answers. Thus, compared with individual edges, edge bundles better enable users to see entity coalitions (visually as grouped edges, connecting with groups of entities), and use such perceived groupings to further explore coordinated relationships.

<table>
<thead>
<tr>
<th></th>
<th>NR</th>
<th>BR</th>
<th>NS</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average number of shared entities in merged biclusters</td>
<td>1.97</td>
<td>2.13</td>
<td>2.34</td>
<td>2.29</td>
</tr>
<tr>
<td>(total number of shared entities of all merged biclusters / total number of merged biclusters)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: Average number of shared entities in submitted merged biclusters.

Besides the variance, we found that the average number of shared entities in merged biclusters (submitted by participants as answers) among all the four experiment conditions was similar, around 2. The detailed information of this is shown in Table 6.4. The number of merged biclusters with different overlap levels are variant among the four conditions, but the average number of shared entities among them remains similar. This implies that sharing 2 entities may be a threshold for participants to determine merging information from different biclusters together. This threshold is potentially useful for automatically merging biclusters or highlighting information to guide users for further investigation.
6.7.2 Accuracy

For each user task, we designed six expected answers. Tow of them are closed biclusters, and others are merged ones. Three of the merged biclusters consist of two biclusters sharing 1, 2 or 4 entities. The other merged one is from two biclusters without overlaps. For example, two groups of people have similar working experience at IT company, although individuals in the groups may work at different companies (e.g., Google, Microsoft and Facebook).

These designed answers serve as golden answers, which are used to evaluate participant answers. Specifically, we used two rules to determine justified user answers: 1) matching or partially matching expected answers (e.g., a subset of an expected answer), and 2) providing supportive evidence. If a user answer satisfies the two rules, it is considered a justified one.

Two types of supportive evidence are acceptable, listed as follows:

1. Graph connection: If a graph connection evidence is given (e.g., showing that 3 people are connected with 3 locations), then a finding is counted as a justified one.

2. Inferred knowledge with connection evidence: If an inference based explanation with its connection evidence is provided (e.g., explaining that 3 people had working experience at IT companies and showing their connections), then a finding is considered justified.

Under the same entity ordering condition, participants found more justified answers, in both quantity and percentage, using bundles than that without bundles. Table 6.5 summarizes participant answers in all four tasks. Comparing the results of two pairs, (NR, BR) and (NS,
Both number and percentage of justified answers in BR and BS are larger than those in NR and NS, respectively. Moreover, we found a significant effect ($F_{(1,76)} = 17.22, p < .0001$) of representation on justified user answers in the four tasks. These results support H2, which means that bundles lead to participants getting more justified answers.

<table>
<thead>
<tr>
<th></th>
<th>NR</th>
<th>BR</th>
<th>NS</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total justified answers</strong></td>
<td>74</td>
<td>93</td>
<td>72</td>
<td>95</td>
</tr>
<tr>
<td><strong>Total answers</strong></td>
<td>109</td>
<td>116</td>
<td>105</td>
<td>120</td>
</tr>
<tr>
<td><strong>Percent of justified answers</strong></td>
<td>68%</td>
<td>80%</td>
<td>69%</td>
<td>79%</td>
</tr>
</tbody>
</table>

Table 6.5: A summary of justified answers in all four tasks

Comparing results of tasks with the same representation but using different orderings, (NR, NS) and (BR, BS), justified user answers do not vary much in both quantity and percentage. In addition, we found an insignificant effect ($F_{(1,76)} = 2.12, p = 0.15$) of entity ordering on justified answers in the four tasks. These results are against H3, which indicates that under the same representation condition, entity orders do not significantly impact participants getting justified findings.

### 6.7.3 Interaction Effort and Time Cost

We use *entity visit* to evaluate interaction effort, which refers to users interacting with entities (e.g., hovering and selecting). Two metrics are applied to measure entity visit: on average, 1) *total number of entities visited* per justified answer, and 2) *total count of entity visits* per
justified answer. Results of the two measures in all four tasks are shown in Figure 6.3 and Figure 6.4 (note: the position of x indicates the mean), respectively.

**Total Number of Visited Entities**

Under the same entity ordering condition, participants visited less number of entities to get a justified answer, when using bundles than that without them. Based on Figure 6.3, comparing the results of two pairs, \((NR, BR)\) and \((NS, BS)\), we can see that the number of visited entities for each justified answer is less with edge bundles. We also found a significant impact \((F_{(1,76)} = 10.52, p < .005)\) of representations on the number of visited entities for justified answers. Considering the two aspects, using bundles significantly reduces the number of visited entities for justified answers, with the same entity orders. Bundles help to reveal entity groupings, so participants do not have to manually investigate this, which may require repetitious entity connections exploration. Thus, they investigated less number of entities to find justified answers.

Similarly, under the same representation condition, seriated ordering reduces the number of visited entities for justified answers. Entity orderings have significant impact \((F_{(1,76)} = 21.14, p < .0001)\) on the number of visited entities for justified answers. Moreover, comparing the results of two pairs, \((NR, NS)\) and \((BR, BS)\), we can find that participants visited less number of entities to get justified answers, with seriated ordering than random ordering. These results indicate that seriated ordering significantly reduces the number of visited entities for justified answers, under the same representation condition. Seriated ordering
places entities, associated with the same bicluster(s), spatially close to each other. With such layout, participants are more likely to find related entities by exploring entities around previously investigated ones. However, using random ordering, participants may have to search and check more entities before they identify useful ones for grouping. This explains the less number of visited entities with seriated ordering.

**Total Count of Entity Visits**

Edge bundles help to reduce total count of entity visits for justified answers, under the same ordering condition. We found a significant impact ($F_{(1,76)} = 6.73, p < .05$) of representation on total count of entity visits for justified answers. In Figure 6.4, comparing two paris, (NR,
BR) and (NS, BS), we can see that the range of average total count of entity visits per justified answer (across all participants) is smaller in BR and BS than NR and NS, respectively. In addition, the majority of data in BR and BS (specifically, the third quantile) is not far above the median (the line inside each box) of NR and NS. This indicates that with bundles, it is more likely that participants took less total count of entity visits to get a justified answer.

![Figure 6.4: A summary of total counts of entity visits per justified answer on average.](image)

Entity ordering does not significantly impact ($F_{(1,76)} = 0.59, p = 0.45 > 0.1$) total count of entity visits for justified answers. In consistent with this, comparing two pairs, (NR, NS) and (BR, BS) in Figure 6.4, the size and position of their corresponding boxes are similar. Thus, considering the two aspects, under the same representation condition, seriated ordering does not significantly reduce total count of entity visits for participants to get justified answers.
Considering both total number of entities visited and total count of entity visits, representation has significant impact, as discussed above. This supports $H_4$, which means that using bundles lead to less entity visits for participants to get justified answers, under the same ordering condition. However, entity ordering has significant impact on the number of entities visited, but not total count of entity visits. If both aspects are considered, $H_5$ is partially supported. This implies that an organized entity order is benefit for information foraging, specifically helping to reduce the effort to find related entities.

**Time Cost**

The time costs per justified answer for different experiment conditions are similar. We found insignificant impact on the time cost for both representation ($F_{(1,76)} = 1.52, p = 0.22 > 0.1$) and entity ordering ($F_{(1,76)} = 0.31, p = 0.58 > 0.1$). A summary of average time (seconds) per justified answer for the four tasks is presented in Figure 6.5. Although there are some variants in this plot (e.g., differences of mean, shown as x), the box plots, corresponding to the four conditions, are similar. These results are against $H_6$ and $H_7$. Thus, neither representation nor entity ordering helps to reduce the time cost for finding justified answers. Layout interpretation may explain this. Although bundles reduce entity visits by revealing entity coalitions, participants still need enough time to understand the computed groupings (e.g., the category of companies). Such interpretation helps them to further decide entities as findings. Moreover, people tend to organize similar information spatially near each other [5]. Seriated ordering organizes entities based on graphic connections. After seriation, entities
spatially near each other are those with similar connections. However, similarity determined by graph connection may not always be consistent with that based on semantic. If the two are conflicted, it may take participants some time to interpret relations of entities associated with bundles, before finally grouping some entities.

### 6.7.4 Summary of Performance Results

In summary, three of the seven hypotheses ($H1$, $H2$ and $H4$) are supported, and one hypothesis, $H5$, is partially supported. Based on this, edge bundles potentially help to direct user attention to closed biclusters and those with higher level of overlaps ($H1$), such as sharing 4 or 2 entities. Moreover, with the same entity ordering condition, bundles significantly
reduce entity visits ($H2$ and $H4$) for participants to get justified answers. Other hypotheses are not supported. Besides reducing the number of visited entities ($H5$), entity ordering has insignificant impact on user performance of exploring coordinated relationships, from the perspective of answer accuracy and the time cost. Although both bundling and seriated ordering can reduce the number of visited entities, they do not significantly impact the time cost for participants to find justified answers. This implies that the interaction effort of entity visits contributes to part of the time cost. Other factors (e.g., layout interpretation) may also impact the time cost for justified answers.

6.7.5 Four Key Trade-offs

Besides performance oriented findings, we identify four trade-offs, considering the two factors: representation and orders.

**Representation Simplicity vs. Task Complexity**

Subjective judgement of task complexity is not always consistent with representation simplicity. Edge bundles potentially reduce visual clutter by aggregating edges. Seriated ordering organizes similar information spatially close to each other. They both attempt to achieve more clear representations from a perspective of simplicity. Using them, total number of edges displayed decreases and the number of edge crossings is reduced. Following this rationale, $BS$ is the easiest condition, while $NR$ is the most difficult one. Both bundles and
seriated ordering are applied in BS, but none of them are used in NR. However, not all participants agreed with this. Based on the interview feedback, in fact, 7 of the 20 participants were in favor of this, although 17 participants considered that tasks with edge bundles were easier than those without bundles.

For participants who were against or partially agreed with the task complexity discussed above, the majority of them (over 50%) voted BR as the easiest condition and considered NS as the hardest one. They thought that there were less edges in BS than that in BR and the same case with NS and NR. This indicates that under the same representation condition, seriated ordering leads to a perceptual illusion on less number of edges. In fact, all four tasks have the exactly same number of edges. These participants thought it were more difficult for them to group entities with less edges. For example, P6 said, “...[compared with BR], there are less edges [in BS], so it is harder for me to make decisions [on grouping entities]...”, and P11 mentioned, “...[compared with NR], the graph [in NS] is sparse [with less connections]. It is more difficult to work with a sparse graph...” The claim, “less edges (or connections)”, indicates that those participants thought the representation got simpler. However, for them, less number of connections means a less likelihood to find possible answers.

Simplicity at the representation level is not always consistent with the perceived complexity of user tasks. Organizing information (in lists) in a meaningful way can lead to a simpler view with less edge crossings. However, attempts for such simple may cause a perceptual misjudgement of the representation, which further impacts user perceived task complexity. Thus, simple representations may sometimes lead to an increase of perceived user task
complexity.

**Connection based Similarity vs. Semantic oriented Similarity**

Participants tend to be misled by spatial proximity of entities and further make unreasonable decisions. One common type of unreasonably justified findings is simply merging entities spatially close to each other. Half of the 20 participants submitted an answer in NS, shown in Figure 6.6, which aggregated two groups of entities on the bottom of two lists. When reviewing this answer, they could not give a reasonable explanation (from a semantic perspective). One popular explanation from them is “...they are near each other, and [when hovering Eric King] they all highlight...” This indicates that they did not pay much attention to these entity labels for getting this finding, which may result from the spatial proximity of these entities. Participants can perceive such spatial proximity. For example, P5 stated, “...entity orders [in NS] seems telling me something...[when hovering entities], highlight ones are almost aligned horizontally.” Based on this, P5 got this answer. However, in NR, for the same group of entities, only 3 participants chose them as a finding, which is much smaller than 10 participants in NS.

Seriated ordering leads to a layout, where spatial proximity reveals graph connection oriented similarity. Entities near each other are potentially associated with the same bicluster(s), and these biclusters, if there are multiple, may reveal different meanings. Due to this, layouts with seriated ordering cannot map similarity between entities, at the semantic level, to their spatial proximity. However, the semantic-level similarity is a key factor that may impact
Figure 6.6: An example of unreasonably justified answers in NS and BS. The selected entities are those inside the blue dotted boxes.

how users organize information. If entity similarities at the two levels are not consistent with each other, for tasks in this study, uses may get unreasonable answers.

It is not trivial to simultaneously reflect entity similarity at both connection level and semantic level by only using spatial proximity. Edge bundles can help, because they visually reveal groupings. Corresponding to the unreasonable finding in NS, for the same group entities in BS, shown in Figure 6.6, two of the 20 participant picked them as a finding. Groupings, revealed as bundles, may lead participants to consider the meaning of entity labels. This helps to avoid simply merging entities, spatially close to each other, together.
**Connectedness vs. Coordinatedness**

Participants show different perception emphasis for different representations. In the view without bundles (NR and NS), participants tend to emphasize the perceived *connectedness* of entities; while using the representation with bundles (BR and BS), they tend to perceive the *coordinatedness* of entities. *Connectedness* means how entities are overall connected, such as strong, week, or isolated. We observed this as participants explained their findings. Figure 6.7 shows an example of two findings with different perception emphasis when participants explained them. Highlighted entities with blue boxes in the left list and those with black border in the right list are selected information in the answers.

Compared with individual edges, edge bundles reveal the *coordinatedness* of entities at the cost of their perceived *connectedness*. When explaining their findings, participants tended to use words, “*strongest or stronger connections*”, in the representation without edge bundles. This conveys their perceived entity connectedness. For instance, P11 explained the finding shown in Figure 6.7 (A), as “*...this group shows the strongest connection between three people with research university and big IT company...*” Actually 9 participants mentioned these words when explaining their findings in NR and NS, but none of them were mentioned in BR and BS. In BR and BS, 7 of the 9 participants changed to use number to address their perceived entity coordinatedness. For example, P12, P17 and P20 explained the answer shown in Figure 6.7 (B), as “*...they [three selected people] all visited two of the three Disney parks...*” However, no participants used number to explain their answers in NR and NS.
Figure 6.7: An example of findings with different perception emphasis in their explanations. (A) shows an answer in NR, explained with emphasis on entity connection. (B) presents a finding in BR, with a coordinatedness oriented explanation.

Comparing the two ways of explanations, a traditional list view helps participants to perceive entity connection. In a view with edge bundles, it is easier for participants to learn entity coordinatedness. This trade-off indicates that edge bundles are better for coordinatedness oriented tasks (e.g., finding three people who all visit four locations), while a list view without bundles better fits connection oriented tasks (e.g., finding the strongest connections between people and locations).
Highlighting Propagation: *Entity*-based vs. *Bundle*-based

Two similar highlighting propagations are applied in the two representations. In the view without bundles, after hovering an entity in a list, connected entities in another list are highlighted. Based on these highlighted entities, all other related entities, in the same list with the hovered one, are also highlighted. In the view with bundles, when hovering an entity, its connected bundles are highlighted, and other entities associated with these bundles are also highlighted. Thus, the former uses an *entity*-based highlighting propagation, while the latter applies a *bundle*-based one. Both attempt to reveal connections between two groups of entities, based on a user hovered entity.

The *entity*-based highlighting propagation leads to user physical interactions during their analysis process. Specifically, seven participants in NR and NS used their fingers to point at some entities on the screen. Three of them used more than one finger to point at two or three highlighted entities, while others used one finger, pointing at a selected entity, and moved the mouse pointer following the edges from this entity. The latter indicates that they tried to check detailed connections from a selected entity, and they used one finger as an additional marker for this selected entity. If multiple entities are selected, particularly when they are close to each other, additional marker may be needed, since they are all highlighted with the same visual encoding. The former, using multiple fingers, indicates that participants attempted to explore *coordinatedness* of highlighted entities. *Entity*-based highlighting propagation cannot clearly reveal the exact coordinated connections, so participants have to manually investigate them. However, if they selected the entities that were
pointed by their fingers, additional entities would be highlighted. In this case, they would lose the current view of highlighted entities. Thus, they used fingers to help remember these entities for further exploration.

Such physical interactions were not observed in the view with the bundle-based highlighting propagation. This suggests that bundles may help participants remember a group of entities to support their analysis. Six participants used the word, “power strip”, to depict the role of edge bundles, which helped them to find and retrieve a group of connected entities from two lists. Thus, even multiple groups of entities are highlighted, participant can use bundles to distinguish different groups. Since bundles work as additional visual markers, so they did not use fingers in their analysis process in BR and BS. Considering user physical interactions, for the view with entity-based highlighting propagation, additional visual markers or extra highlighting mechanism may be needed.

6.8 Discussion

We evaluate two key bicluster-based techniques of BiSet: edge bundling and entity ordering. Under the same ordering condition, compared with no bundles, using edge bundles leads to more user justified findings with less interaction effort. With the same representation, seriated ordering requires less interaction effort for getting a justified answer. We identified four key trade-offs considering the two techniques. These results indicate that for supporting coordinated relationship exploration, bicluster-based edge bundling is critical.
6.8.1 Considering the Role of Machine Learning

As is shown in the study, complex tasks (e.g., exploring coordinated relationships) may not be directly solved by machine learning results (e.g., biclusters), and human effort is still needed. For example, participants merged information from different biclusters to get an answer. Although computed biclusters reveal advanced patterns of data, they cannot cover all answers of complex user tasks. However, such computed entity coalitions can still benefit users in coordinated relationship exploration, by enabling them to see entity groupings as edge bundles and reducing their entity visits to find answers.

Besides finding patterns at data level, another key role of machine learning for visual analytics is to free human from low-level perceptual problems and support making high level inferences. In BiSet, individual edges are bundled based on computed biclusters, which potentially applies machine learning at the representation level. Such bundles reduce participant effort of entity visits to find coordinated relationships and increase their justified answers. Moreover, the bundles help users to overcome perceptual problems caused by spatial proximity of entities. This not only helps participants to avoid making wrong judgements that are misled by spatial closeness, but also leads them to consider the meaning of entities labels for high level inference (e.g., identifying a group students as HCI students based on the courses taken). The two benefits at data level and representation level make it critical to use bicluster-based edge bundling for support coordinated relationships exploration.
6.8.2 Implication for Tools of Visualizing Relations

Edge bundling is critical for applications that visualize relationships to support sensemaking tasks (e.g., Jigsaw’s list view), particularly for coordinated relationships. Individual edges show basic relations between entities, but they lack the capability to reveal entity coalitions (e.g., connected groups of entities) due to missing visual markers to reveal groupings. Although highlighting can help to reveal entity groupings, it cannot easily serve as a handler that enables direct manipulation on a group of entities (e.g., dragging and moving them in space). Based on study results, besides enabling users to see entity groupings, edge bundles also help them overcome perception problems and make high level inference. Moreover, edge bundling can be applied to variant layouts, not limited to lists, so it is flexible to be used to reveal groupings between different types of visualizations (e.g., connecting locations in a map and points in a scatterplot). Thus, in order to support user exploration of complex relations, a data-driven edge bundling (e.g., based on biclusters) offers a good solution.

The identified trade-offs indicate three key considerations to apply BiSet for coordinated relationships exploration: 1) user controlled dynamic ordering, 2) task driven layout selection, and 3) coordinatedness highlight. Instead of using static orders, enabling users to dynamically organize entities and biclusters are useful to support their analysis, since a simple layout is not always lead to a perceived simple task. Besides this, it is hard to encode both semantic similarity and connection oriented similarity with just one spatial order, so dynamic ordering potentially allows users to explore data from different perspectives. Moreover, different representations may fit different user tasks. For coordinatedness oriented tasks, edge bundles
work better, while for connectedness oriented tasks, a representation without bundles may lead to better results. However, if users have to deal with both types of tasks, enabling them switch representations potentially offers a good solution. In this case, we may consider adding coordinatedness oriented highlighting to the view without bundles, or enabling users to switch from individual edges to bundles in a local manner (for a group of entities under investigation), and vice versa. This helps to reveal both coordinatedness and connectedness of entities, without costing extra physical interactions (e.g., pointing at entities with fingers).

6.8.3 Study Limitation

Although the identified trade-offs indicate considerations to apply BiSet for exploring coordinated relationships, we admit one limitation of this study.

The familiarity with entity labels may potentially impact the study result. Although task orders were shuffled in the study, we did not change the combination between datasets and experiment conditions. This results in a fixed task domain for each experiment condition, listed in Table 6.3. Each participant was assigned all four tasks with a random order, but they may be more familiar with labels in one dataset than those in another. In the study, we did not measure user familiarity with entity labels, so it is not clear whether this would also impact user performance of the assigned tasks. In order to gain a better understanding the impact of familiarity with entity labels on user performance in exploring coordinated relationships, a further exploration is needed.
Chapter 7

Conclusion

Bicluster, a bundled relations between two sets of entities, potentially reveals coordinated relationships from data. It also offers a conceptual format to present coordinated relationships in an organized manner. With it, it is possible to present coordinated relationships in a usable way, and further support user exploration for sensemaking tasks.

This work is driven by the following research questions:

RQ1: What is the design space of bicluster visualizations?

(a) How can a design framework be built to inform the design of bicluster visualizations?

(b) What is the key design trade-off of bicluster visualizations?

RQ2: How can we instantiate the identified design options?

(a) How can biclusters be visualized based on the proposed design framework?
(b) What interactions can be applied to the visualized biclusters to support user exploration of coordinated relationships?

**RQ3**: How does the proposed bicluster visualization impact users’ exploration of coordinated relationships?

(a) Do bicluster oriented edge bundling and entity ordering, compared with traditional list representations, help users more efficiently explore coordinated relations?

(b) Comparing a traditional list view with BiSet, what are the trade-offs when using them for coordinated relationships exploration?

### 7.1 Research Contributions

**A five-level Design Framework**

In this work, we present a five-level design framework for bicluster visualizations (for RQ1(a)), with a survey of the relevant state-of-the-art design considerations and applications. We summarize pros and cons of these design options for supporting user tasks at each of the five levels. This framework works as systematic design guidelines for bicluster visualizations.

**BiSet**

Based on the framework, we identify the key design trade-off to visualize biclusters: entity-centric versus relationship-centric (for RQ1(b)). To balance this trade-off, we propose
the BiSet technique, which displays biclusters as edge bundles in context as bundled edges between sets of related entities (for RQ2(a)) . We make bundles as the first class objects and add a new layer “in-between” lists to contain these bundle objects. BiSet enables user interactions on both edge bundles and entities (for RQ2(b)) for organizing relevant information in a bidirectional way. Users can interact with edge bundles to forage and organize relevant entities and, vice versa, for sensemaking purposes.

**Advanced Interaction to Scale Up**

In order to confront with large datasets, we present four advanced interactions in BiSet to scale it up (for RQ2(b)): *seriation, aggregation, prioritization* and *attraction*. We propose an approach to enable performing seriation in a list-based layout by using merged adjacency matrices, which helps to organize elements in lists and reduce edge crossings in the layout. We present two interactive approaches (automatic and manual) to aggregate biclusters. We enable prioritizing biclusters or bicluster-chains in BiSet to guide user exploration, using an amalgamation of algorithmic and human-driven techniques. Two levels (bicluster-level and entity-level) of dust and magnet visual metaphor [140] are applied in BiSet to enable spatially *attracting* similar biclusters and their related entities, when users drag a bicluster.

**Trade-offs: BiSet vs. a Traditional List View**

The study, presented in this work, evaluates the two bicluster-based techniques in BiSet: *edge bundling* and *seriated ordering*. Results of this study show the effectiveness of the two techniques (RQ3(a)) (e.g., reducing entity visits). In addition, we identify four key trade-offs
(RQ3(b)), comparing BiSet with a traditional list view. These trade-offs imply three important considerations to apply BiSet for exploring coordinated relationships. Furthermore, it indicates two levels of benefit for applying machine learning in visual analytics: 1) identifying advanced patterns at data level, and 2) freeing human from low-level perceptual problems and supporting making high level inferences at representation level.

7.2 Future Opportunities

With algorithm parameters (e.g., size), users can control bicluster discovery. However, there are still usability challenges of interacting with biclusters for visual analytics usage, particularly in three key aspects: 1) algorithm comparison, 2) algorithm design and parameter manipulation, 3) bicluster evaluation. These challenges raise questions about usable biclusters from four different levels: model level, parameter level and evaluation level.

7.2.1 Algorithm Comparison

*How to enable users to reasonably select biclustering algorithm(s)* is the first challenge to make biclusters usable. Different algorithms may use different criterion for bicluster discovery, so different algorithms may find different biclusters for the same dataset. Thus, it is necessary for users to decide which algorithm(s) to use by leveraging algorithmic results with their domain knowledge and analysis tasks. Compared with arbitrary selections, making comparison among different algorithms can better help user decision making. However,
how to compare different biclustering algorithms still remains a question. Specifically, what aspects of biclustering algorithms (e.g., parameters, performance, results, etc.) are usable for comparison? Are these driven by specific user tasks?

7.2.2 Algorithm Design and Parameter Manipulation

Algorithm design and enabling (novel) user interactions to steer algorithms is another challenge of interacting with biclusters. Currently, the procedure of using biclusters for sense-making includes two sequential steps: algorithmic bicluster discovery and user investigation. Figure 7.1 shows the paradigm of this process. This paradigm has been applied in recent visual analytics tools (e.g., BiSet [123], Bixplorer [43], Furby [120], etc.). In these tools, users interact with visual metaphors of biclusters to explore meaningful and useful ones from algorithmic results. However, user reasoning results (e.g., biclusters identified as meaningful ones) cannot be interpreted by the selected biclustering algorithm(s) and further impact the bicluster discovery process in future. Thus, users have to passively “accept” all algorithm results and then do explorations. This limits human investigations to a function of post-clustering filters.

Semantic interaction offers a novel way for users to interact with machine learning algorithms, and it uses (interpret) algorithms to enable the injection of user reasoning into a computational process [38]. Can this concept be applied to inform human-centered biclustering algorithm design? Particularly, in addition to existing algorithm parameters (e.g., schema
and size), what additional parameters can we add to biclustering algorithms to control bicluster discovery? For example, if users identify locations, Boston and Seattle, as useful ones, how can we inform algorithms of this information, and further steer them to find biclusters containing these locations? Different from the sequential paradigm depicted in Figure 7.1, this requires iterative processes. Since user decisions and/or intentions can be inferred by algorithms, the role of user investigations becomes more active to steer algorithms, rather than just post-clustering filters.

How to enable user to manipulate algorithm parameters is another key question. Besides
command line, there are two ways for users to adjust algorithm parameters: user interface widget (e.g., sliders, buttons, spinner, etc.), and direct manipulation (e.g., the “near-similar” metaphor in semantic interaction [38]). The latter is less precise in parameter adjustment than the former. For example, when users drag one node closer to another, they may not know the exact distance between the two nodes. However, when they use a spinner, they can precisely change the distance between the two nodes. For existing biclustering algorithm parameters (e.g., size), user interface widgets may be a good choice due to the precise adjustment capability. For other parameters, possibly identified and added in future human-centered biclustering algorithms discussed above, how can we provide usable ways for users to interact with them?

7.2.3 Bicluster Evaluation

Computationally prioritizing biclusters helps to direct user attention to useful ones. This is useful especially when handling large datasets. However, how to evaluate biclusters and prioritize them both computationally and visually still remains a challenge. Using BiSet and maximum entropy model (MaxEnt), a preliminary attempt has been performed in [137]. In this exploration, biclusters are evaluated using MaxEnt, based on entity distribution. According to the score of each evaluated biclusters, given by MaxEnt, BiSet visually prioritize them with color codings. With a case study, this approach has been reported with promising results, but it still needs users to interpret semantic connections of statistically relevant biclusters. Besides such distribution based evaluation, can biclusters be evaluated based on
semantic meanings or meta data [20]? Moreover, how can we incorporate such computational evaluations into a user reasoning process to support sensemaking progressively, and how can we enable users interactively control these evaluations?

7.2.4 Summary

In summary, these challenges lead to more opportunities of using biclusters for visual analytics. Besides exploring visual representations of biclusters, how to interact with them during the process of bicluster generation may help analysts more effectively explore their data because of two reasons. For one thing, they do not have to manually investigate many visualized biclusters. For another thing, their interactions may impact bicluster generation, which results in meaningful bicluster candidates. However, this involves an iterative human-model interaction procedure for data analytics, which requires further explorations of interacting with biclusters. This work leads a way to display biclusters in a usable manner, so it fundamentally builds a bridge to enable users directly manipulate mined biclusters for coordinated relationships exploration.
Bibliography


[74] J. Johansson, C. Forsell, M. Lind, and M. Cooper. Perceiving Patterns in Parallel


[89] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis:


Appendix

The following are the four synthetic datasets used in the primary study for the four experiment conditions: NR, BR, NS and BS.

<table>
<thead>
<tr>
<th>NR</th>
<th>BR</th>
<th>NS</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The dataset used for the NR condition in the primary study.
Figure 8.2: The dataset used for the BR condition in the primary study.

The dataset used for the BR condition in the primary study.

Figure 8.3: The dataset used for the NS condition in the primary study.
The dataset used for the BS condition in the primary study.