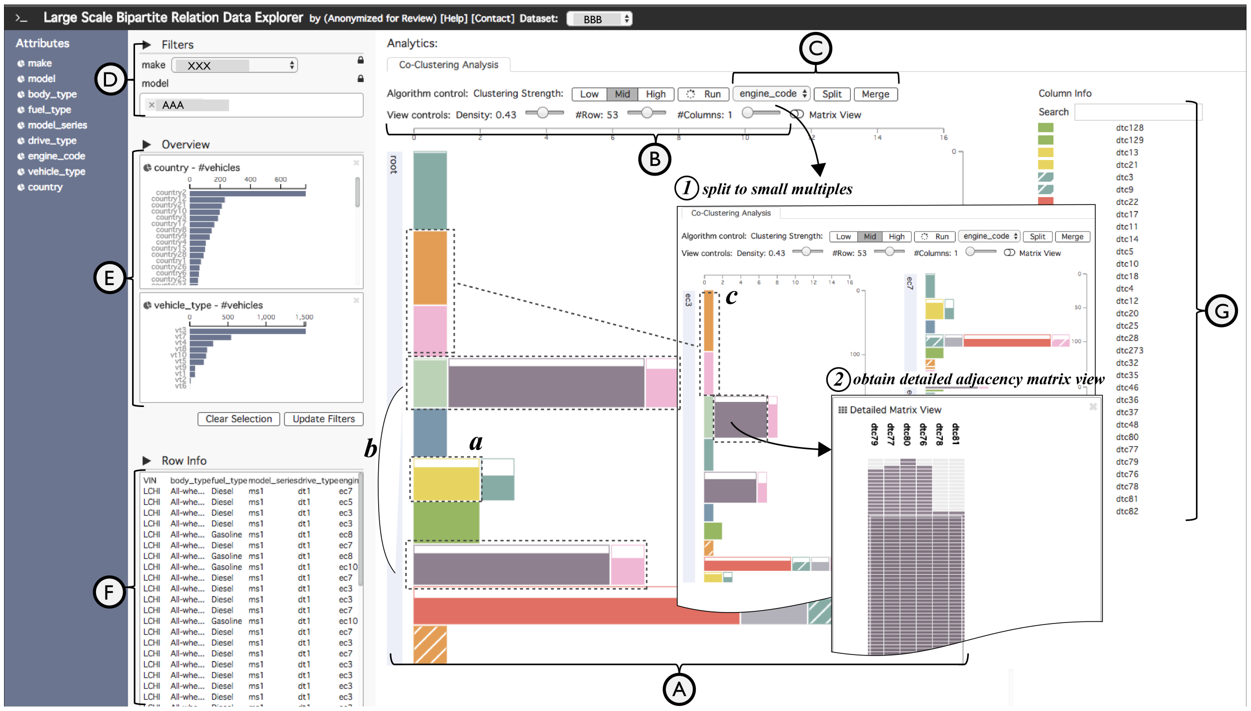
ViBr: Visualizing Bipartite Relations at Scale with the Minimum Description Length Principle

Gromit Yeuk-Yin Chan*, Panpan Xu, Zeng Dai and Liu Ren


Large Scale Bipartite Relation Data Explorer by (Anonymized for Review) [Help] [Contact] Dataset: BBB
Attributes
make
model
body_type
fuel_type
model_series
drive_type
engine_code
vehicle_type
country
Filters
make XXX
model
AAA
Overview
country - #vehicles
vehicle_type - #vehicles
Clear Selection
Update Filters
Row Info
Analytics:
Co-Clustering Analysis
Algorithm control: Clustering Strength: Low Mid High Run engine_code Split Merge
View controls: Density: 0.43 #Row: 53 #Columns: 1 Matrix View
Column Info
Search
1 split to small multiples
2 obtain detailed adjacency matrix view
Detailed Matrix View
a
b
c
A
B
C
D
E
F
G

VIN	body_type	fuel_type	model_series	drive_type	engin
LCHI	All-whe...	Diesel	ms1	dt1	ec7
LCHI	All-whe...	Diesel	ms1	dt1	ec5
LCHI	All-whe...	Diesel	ms1	dt1	ec3
LCHI	All-whe...	Diesel	ms1	dt1	ec3
LCHI	All-whe...	Gasoline	ms1	dt1	ec8
LCHI	All-whe...	Diesel	ms1	dt1	ec7
LCHI	All-whe...	Gasoline	ms1	dt1	ec8
LCHI	All-whe...	Gasoline	ms1	dt1	ec10
LCHI	All-whe...	Diesel	ms1	dt1	ec7
LCHI	All-whe...	Diesel	ms1	dt1	ec3
LCHI	All-whe...	Diesel	ms1	dt1	ec3
LCHI	All-whe...	Diesel	ms1	dt1	ec7
LCHI	All-whe...	Gasoline	ms1	dt1	ec10
LCHI	All-whe...	Diesel	ms1	dt1	ec7
LCHI	All-whe...	Diesel	ms1	dt1	ec3
LCHI	All-whe...	Diesel	ms1	dt1	ec7
LCHI	All-whe...	Diesel	ms1	dt1	ec3
LCHI	All-whe...	Diesel	ms1	dt1	ec3
LCHI	All-whe...	Diesel	ms1	dt1	ec3
LCHI	All-whe...	Diesel	ms1	dt1	ec3

Fig. 1. Interface of ViBr demonstrating how analyst discovers different patterns of vehicle faults of a car model using bipartite graph summarization. First she selects the data using the filters Ⓓ and computes a summarization Ⓐ filtered by the density and the sizes of the clusters Ⓑ. From the adjacency list style overview Ⓐ she observes several interesting groups of vehicles with different fault patterns (***a*** and ***b***). Splitting the summary view into small multiples ①, several unique groups of faults (e.g. ***c***) only occur in vehicles with a particular engine code. She further dives into the next level of detail by bringing up a matrix view for a particular block ②. A reference of labels is always provided in the legend bar with text search provided Ⓖ. The node attribute value distributions and detail node information are displayed in Ⓔ and table Ⓕ respectively.

Abstract—Bipartite graphs model the key relations in many large scale real-world data: customers purchasing items, legislators voting for bills, people's affiliation with different social groups, faults occurring in vehicles, etc. However, it is challenging to visualize large scale bipartite graphs with tens of thousands or even more nodes or edges. In this paper, we propose a novel visual summarization technique for bipartite graphs based on the minimum description length (MDL) principle. The method simultaneously groups the two different set of nodes and constructs aggregated bipartite relations with balanced granularity and precision. It addresses the key trade-off that often occurs for visualizing large scale and noisy data: acquiring a clear and uncluttered overview while maximizing the information content in it. We formulate the visual summarization task as a co-clustering problem and propose an efficient algorithm based on locality sensitive hashing (LSH) that can easily scale to large graphs under reasonable interactive time constraints that previous related methods cannot satisfy. The method leads to the opportunity of introducing a visual analytics framework with multiple levels-of-detail to facilitate interactive data exploration. In the framework, we also introduce a compact visual design inspired by adjacency list representation of graphs as the building block for a small multiples display to compare the bipartite relations for different subsets of data. We showcase the applicability and effectiveness of our approach by applying it on synthetic data with ground truth and performing case studies on real-world datasets from two application domains including roll-call vote record analysis and vehicle fault pattern analysis. Interviews with experts in the political science community and the automotive industry further highlight the benefits of our approach.

Index Terms—Bipartite Graph, Visual Summarization, Minimum Description Length, Information Theory

1 Introduction

Understanding bipartite relations is the key to gain insight from data in a variety of application domains. Such activity can often be seen in

*Work done during internship.

Gromit Yeuk-Yin Chan is with New York University. E-mail: gromit.chan@nyu.edu. Panpan Xu, Zeng Dai and Liu Ren are with Bosch Research North America, Sunnyvale. E-mail: Panpan.Xu, Zeng.Dai, Liu.Ren@us.bosch.com

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

user preferences identification on movie recommender systems [19], market basket analysis on sales records [15] , political leanings analysis on roll-call vote records [4] and relationship discovery in urban open data [6, 8]. As “a picture is worth a thousand words,” visualization plays an important role in landing a good hypothesis or direction for domain experts to analyze bipartite relations at scale.

Recently many visualization techniques have been proposed to support bipartite relation analysis [12, 37, 39, 44, 47]. Nonetheless, the increasing volume and complexity of the data bring new challenges. First, revealing all information at once will exceed human's cognitive ability to conduct effective analysis. In our use cases (Sect. 7) data either contains more than 170, 000 bipartite connections or contains 5,000 - 43,000 nodes depending on the subset selected for analysis. Plainly showing the data will be considered as infeasible. A better way to help analysts start the data exploration is to construct a broad overview of the data instead of showcasing each individual entity and bipartite connection. Second, noises are prevalent in real-world datasets, thus the insights are common to contain artifacts as well. Thus, what analyst needs is a robust visualization technique that reveals the most general and salient patterns in the entire dataset.

To address these challenges, we describe a novel visual summarization technique for bipartite relational data using an *information-theoretic* approach. We apply the Minimum Description Length (MDL) principle [32] which provides a criteria to optimize the aggregation of bipartite relations to create a high-level overview, balancing visual complexity and information loss in the display. We visually represent the original data with the aggregated bipartite connections, and in the meanwhile model the information loss with the corrections needed to recover the original data from the aggregated graph.

Using the MDL principle in visual data summarization has been seen in hierarchical data [42] and event sequence data [7], but applying it to large scale bipartite relation data imposes new challenges and opportunities. Apart from formulating the principle for bipartite relations, we also describe how to speed it up with locality sensitive hashing (LSH) [22], which effectively improves the running time without significantly degrading the results. We further introduce a tailored and space efficient visualization design inspired by visual adjacency lists [18] to display the aggregated bipartite relations. The compact visual design fits nicely in a small multiples display for visual comparison of bipartite relations across different subsets of data, which can be created by faceting on a selected node attribute. A comprehensive visual analytic system is developed as well to support data exploration at varying levels-of-detail, correlating domain-specific node attributes with the relation patterns, and filtering and selecting subsets for focused analysis to cope with the challenges in usabilities [24]. In short, our contributions are as follows:

- We apply the MDL principle to pack large scale and noisy bipartite relation data into a highly compressed representation which is suitable for a coarse-level overview of the data.
- We propose an efficient algorithm based on LSH to optimize the MDL optimization process to facilitate interactive analysis.
- We introduce novel visual analytics techniques and interaction designs for exploring large scale multivariate bipartite graph data.
- We present two example usage scenarios with real-world datasets and domain specific analytic tasks that demonstrate the usability, effectiveness and general applicability of our technique.

2 Related Work

Bipartite graph exists in many application domains and a variety of visualization techniques have been developed in the past. Here we categorize the related work into plain bipartite graph visualization and bipartite graph aggregation for a high-level overview of the data.

2.1 Bipartite relation visualization

The most common approach to visualize bipartite relations is to position two sets of nodes on separated regions on the display and draw edges as (curved) lines between the nodes. Related techniques can be seen in various visualizations including semantic substrates [34], PivotPath [10], Jigsaw [36], and parallel node-link bands [13]. All of them applied such principle to display bipartite relation and encode additional attributes with node sizes and colors, edge widths, opacities and coloring, etc. Another layout style for bipartite graph is unimodal, which treats the bipartite graph as a whole without spatial separation. Such visualization techniques use color, shape or other visual channels to distinguish the sets, which can be seen in FacetAtlas [3], OntoVis [33] and Anchored Maps [26]. Besides bipartite graphs can also be represented by adjacency matrices. The visualization techniques usually build upon the simplest form of adjacency matrices and enhance it with additional visual cues or interactive functionalities [20,35]. Row and column seriation techniques further facilitate pattern recognition in matrix displays [2, 31, 35]. Besides node-link diagrams and adjacency matrices, bipartite graphs can also be visually represented by a hybrid of the two, highlighting the bi-cluster structure detected by automatic algorithms as in Bixplorer [12], or visualizing additional topological structures on top of the bipartite relations [27].

Bipartite graphs can also be generally represented as sets covered in a recent survey about the state-of-the-art of set visualization techniques [1]. In general, most of the techniques visualize a moderately sized dataset with at most hundreds of sets/items. Our work focuses on providing a concise overview of large scale bipartite graphs with tens of thousands or even more nodes and edges.

2.2 Bipartite relation clustering and visualization

To support scalable analysis and pattern detection on bipartite graph data, work has been focusing on graph summarization through edge or node aggregation. We group the graph summarization algorithms and the corresponding visualization techniques into two major categories:

Algorithms including CHARM [45] and LCM [41] extracts bi-cliques in coordinated bipartite relations. Bixplorer [12, 38, 40], BiSet [39], and BiDots [47] utilize them to identify and visualize bi-cliques. The visualization techniques display such structures using overlays on top of matrices/node-link diagrams, bundles edges within a bi-clique in node-link diagrams [23, 30, 39], or combine both to form a hybrid representation [12, 37]. However, real-world data is usually noisy and missing links may create a lot of fragmented bi-cliques that overwhelm the users. Although interactive exploratory visualization techniques can partly alleviate this issue [39,47], a high-level overview still cannot be obtained easily.

Another category of bipartite graph clustering algorithms simultaneously group the nodes in the two partitions, relaxing the requirements on bi-cliques. Spectral co-clustering [21] and spectral bi-clustering [9] were classical methods. Recently, some bipartite visualization techniques utilize those algorithms to perform data aggregation and reduce visual clutter in the display, including Xu *et al.* [44] and Ming *et al.* [25].

VIBR falls into this research domain, although we formulate the co-clustering problem with the information theoretic minimum description length principle (MDL) [32], which allows the method to directly quantify and minimize the information loss in the visual display. Recently, Veras and Collins [42] and Chen *et al.* [7] apply it to construct high-level visual summary of hierarchical and temporal event sequence data respectively to balance the conciseness and information content in visualization. The MDL principle has also been applied to graph summarization in the database and data mining research community. Navlakha *et al.* [29] is the first work proposing summarization of general graphs with bounded error based on a two-part representation of summary graph and corrections.

A similar approach (SCMiner) has been proposed by Feng *et al.* [11] for weighted bipartite graphs. However, our algorithm is accelerated with LSH which experimentally reveals that our algorithm has over 300 times speed gain as compared to SCMiner and three times speed gain compared to Cross-association [5] (a matrix clustering algorithm), making it more suitable for interactive exploration of data.

We further propose a novel design to visualize the aggregation inspired by the visual adjacency list design for dynamic graph visualization [18]. The visual design provides a compact overview of the bipartite relations and facilitates visual comparison across different subsets in the data. The analysts can facet on a selected node attribute and compare the bipartite connections.

3 Minimum Description Length (MDL) for Bipartite Graph Summarization

In the following discussion we use U and V to denote the two sets of nodes and the bipartite graph is $R \subseteq U \times V$. In this section we first introduce a two-part representation of a bipartite graph, inspired by the two-part representation of general graphs in [29], which consists of a summary graph and a set of residual edges. Combining it with the MDL principle we formulate an optimization goal to identify a simultaneous grouping (i.e., co-clustering) of U and V such that the corresponding summary graph can describe the original data balancing complexity and information loss.

3.1 Two-part representation of a bipartite graph

The two-part representation is illustrated in Fig. 2. Given a simultaneous grouping of the nodes in U and V (Fig. 2(a)), it consists of:

A summary graph $\mathbb{S}$ with the meta-nodes and their interconnections as illustrated in Fig. 2(b). An edge is created between two meta-nodes if the connection is dense in the original graph. For example, it is almost a bi-clique between $\{1,2,3\}$ and $\{a,b,c\}$, therefore in the summary graph an edge is created between these two meta-nodes. On the other hand only one edge exists in the original graph between $\{1,2,3\}$ and $\{d\}$, so the summary graph between these two has no edges.

A list of corrections $\mathbb{C}$ that can recreate the original data from the summary graph. The summary graph is an approximate representation of the original data. With the meta-edges we can infer that the interconnection between two clusters is dense. However it is not enough to recover the exact bipartite connections. We need to include additional corrections to remove the non-existing edges. For example, between $\{1,2,3\}$ and $\{a,b,c\}$, every edge exists except for $(2,c)$, therefore we add an additional correction to remove $(2,c)$ which does not exist in the original graph (Fig. 2(c)). On the other hand, even when meta-edges do not exist in the summary graph, it is still possible for some edges to appear in the original data, therefore another type of correction add edges back. For example, between $\{1,2,3\}$ and $\{d\}$ no edges exist except for $(1,d)$, so we add back $(1,d)$ (Fig. 2(c)).

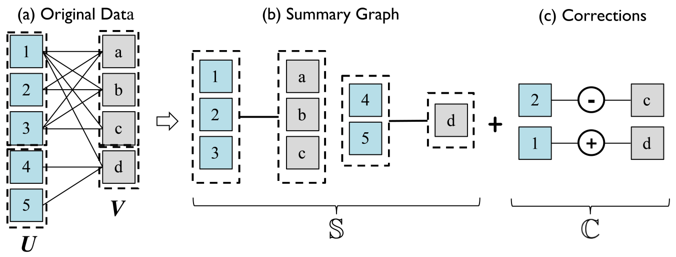


Fig. 2. A bipartite graph can be represented as a summary graph $\mathbb{S}$ with corrections $\mathbb{C}$. The original set requires 11 units of spaces (11 edges) while $\mathbb{S}$ and $\mathbb{C}$ together only require 4 (2 in $\mathbb{S}$ and 2 in $\mathbb{C}$).

Combining the summary graph and the corrections we can fully recover the original graph. The two-part representation is therefore a *lossless* representation of the original data. The summary graph $\mathbb{S}$ can provide a coarse-level overview of the data and the corrections $\mathbb{C}$ model the information loss in the display. The visual complexity dramatically decreases in the overview and user can immediately grasp the dominant connectivity patterns in the bipartite graph. Visual abstraction of the data is even more critical for understanding bipartite relations with thousands or even millions of nodes and edges when it becomes almost impossible to fit all the raw data on a single screen.

The remaining problem is how to identify a summary graph which can best represent the underlying data balancing the visual complexity and information loss. This problem eventually boils down to identifying an optimal grouping of the nodes in U and V based on which we can bundle the edges to form the summary graph.

3.2 The MDL principle

We propose an algorithm to obtain an optimal grouping of the nodes in U and V simultaneously following the minimum description length (MDL) principle. The MDL principle states that the best model (or hypothesis) of a dataset should minimize its total description length L, which consists of the model description length and the description length of the original data with the help of the model:

$$L = L(M) + L(D|M)$$

For a bipartite graph, the model is the summary graph $\mathbb{S}$ and given a summary graph we can use the corresponding corrections part $\mathbb{C}$ to recover the original data. Our goal is to obtain an optimal grouping of the nodes such that it can minimize the total description length of the summary graph and the corrections. To state it more formally, we denote a bipartite graph as $R \subseteq U \times V$. Our goal is to identify a partition of U and V such that it can minimize the total description length:

$$L_R(P,Q) = L(\mathbb{S}) + L(\mathbb{C})$$

where P is a partition of U, Q is a partition of V, $\mathbb{S} \subseteq P \times Q$ is the summary graph and $\mathbb{C}$ is the set of corrections. The definition of $\mathbb{C}$ is:

$$\mathbb{C} = (\cup_{(p,q)\in\mathbb{S}} p \times q) \oplus R$$

where $\oplus$ denotes the disjunctive union between sets. Since we only need to store the meta-edge information for the summary graph, the description length is $L(\mathbb{S}) \propto \|\mathbb{S}\|$ and the description length of the corrections is $L(\mathbb{C}) \propto \|\mathbb{C}\|$. We further introduce the parameters α, β_P and β_Q to control the penalty of the corrections and the number of clusters, similar to Veras and Colins *et al.* [42] and Chen *et al.* [7]. To sum up, our goal is to find P and Q that can minimize the loss function:

$$L_R(P,Q) = \|\mathbb{S}\| + \alpha\|(\cup_{(p,q)\in\mathbb{S}} p \times q) \oplus R\| + \beta_P\|P\| + \beta_Q\|Q\| \quad (1)$$

where $\beta_P\|P\| + \beta_Q\|Q\|$ can be considered as two regularization terms which penalize large number of node clusters. Larger β_P and β_Q results in smaller numbers of clusters. An example of the effect can be found in **??** (Appendix).

4 Computing MDL Representation

In this section, we first introduce a basic algorithm to find partition P and Q and the corresponding summary graph $\mathbb{S}$ that can minimize the description cost described in Equation 1. Then we describe a speed up strategy that applies LSH [22], an efficient nearest neighbor search algorithm. We report the results of a series of empirical experiments to verify the robustness of the algorithm and compare it with other co-clustering algorithms.

4.1 The *BM-MDL* algorithm

We first propose a basic version of the algorithm named *BM-MDL* (bipartite graph mining with MDL) based on the approach proposed by Navlakha *et al.* [29]. The algorithm follows a bottom-up and greedy approach. Initially each node is treated as an individual cluster. In each iteration, we identify a pair of clusters to merge that will result in the maximum reduction in description length. The process stops when the total description length no longer decreases. As a simple speed up strategy, we use a randomized approach which picks a cluster randomly and merge it with the best node in its hop-2 neighborhood, similar to Navlakha *et al.* [29]. For example, in Fig. 2, if node 1 is first chosen, the algorithm will try to merge it with its 2-hop neighbors including node 2, 3, 4 and 5. Merging node 1 and 2 creates two meta-edges $(\{1,2\},\{a\})$ and $(\{1,2\},\{b\})$ in $\mathbb{S}$ and two additional correction edges $(1,c)$ and $(1,d)$ in $\mathbb{C}$. Assuming $\alpha = 1$, $\Delta L = 2 + \beta_P$ since there are two edges less in total and the number of node clusters in P reduces by one. Similarly, the algorithm calculates ΔL by merging node 1 with node 3, 4 or 5, this result in $\Delta L = 3, 1, 1$ respectively with an additional constant β_P. The algorithm therefore will choose node 3 and merge it with node 1. The procedure is described in detail in Algorithm 1. The subroutine *cost_reduction_for_bundling* in line 10 calculates the change in description length by merging two meta-edges in the summary graph, which is a necessary step for merging two meta-nodes. The subroutine *merge* in line 18 updates the partitions P or Q and the

Algorithm 1: BM-MDL

```
Input: Two sets of nodes $U$ and $V$ and the bipartite relation
       $R \subseteq U \times V$
Output: Partition of $U$, denoted as $P$ and partition of $V$, denoted
        as $Q$, summary graph $\mathbb{S} \subseteq P \times Q$
/* Initialization step */
$P = \{\{u\}|u \in U\}$; $Q = \{\{v\}|v \in V\}$
$\mathbb{S} = \{(\{u\},\{v\})|(u,v) \in R\}$
$\Delta L_{max} = 1$
/* Iterative merging step, until no cost reduction
   is possible */
while $\Delta L_{max} > 0$ do
    /* Merge clusters in $P$ */
    $p_0 = random\_select(P)$
    $\Delta L_{max} = 0$, $p_{max} = undefined$
    for $p \in two\_hop\_neighbors(p_0, \mathbb{S})$ do
        $\Delta L = 0$
        for $q \in neighbors(p,\mathbb{S}) \cup neighbors(p_0,\mathbb{S})$ do
            $\Delta L+=$
              $cost\_reduction\_for\_bundling((p,q),(p_0,q),R,\mathbb{S})$
        end
        $\Delta L+ = \beta_P$
        if $\Delta L > \Delta L_{max}$ then
            $\Delta L_{max} = \Delta L$, $p_{max} = p$
        end
    end
    /* Merge two clusters if cost reduction is
       possible */
    if $\Delta L_{max} > 0$ then
        $merge(p_0, p_{max}, R, \mathbb{S})$
    end
    /* Same procedure as for $Q$... */
end
return $P$, $Q$, and $\mathbb{S}$
```

summary graph $\mathbb{S}$ by merging two meta-nodes. In Appendix we provide more detail on the two subroutines *cost_reduction_for_bundling* and *merge*. In each iteration, the algorithm computes the description length reduction for all the hop-2 neighbors of a node. Assuming that the nodes have average degree of d, $O(d^2)$ hop-2 candidate pairs have to be checked for each iteration [29].

4.2 Speeding up With LSH

The basic version of the algorithm is extremely time consuming due to the need to compute and compare the potential cost reductions for merging each pair of clusters with 2-hops in the bipartite graph. To speed up the algorithm, we employ locality sensitive hashing (LSH) [22], which is an efficient method for nearest neighbor search. We use LSH to efficiently identify the clusters with the most similar bipartite connections measured by Jaccard similarity. The procedure is described in Algorithm 2. Since LSH allows very efficient search for nodes with similar bipartite connections, the number of candidate pairs to check is much less than $O(d^2)$ [22].

Notice that the inner while loop in Algorithm 2 is similar to Algorithm 1 except that now instead of identifying the best clusters to merge in the hop-2 neighbors, we only search among those pairs of clusters with Jaccard similarity coefficient above a certain threshold θ (line 12) which can be efficiently done approximately with LSH [22]. Using the same example in Fig. 2 and above, with a proper setting of θ, if node 1 is first chosen, the algorithm will compare with node 3 only since they have the greatest similarity of connecting edges, eventually 1 and 3 will be merged as the description length will decrease. The outer loop sets θ at a relatively high value (close to 1.0 as in line 4) initially and gradually decrease it by a fixed decay rate λ. This allows the algorithm to prioritize the most similar clusters in early iterations but still being exhaustive in the search at later stages.

Algorithm 2: BM-MDL-LSH

```
Input: Two sets of nodes $U$ and $V$ and the bipartite relation
       $R \subseteq U \times V$
Output: Partition of $U$, denoted as $P$ and partition of $V$, denoted
        as $Q$
/* Initialization step */
$P = \{\{u\}|u \in U\}$; $Q = \{\{v\}|v \in V\}$
$\mathbb{S} = \{(\{u\},\{v\})|(u,v) \in R\}$
$\Delta L_{max} = 1$
$\theta = 0.99$, $\theta_{cutoff} = 0.1$, $\lambda_{decay} = 0.9$
/* Iterative merging step */
while $\theta > \theta_{cutoff}$ do
    $T_P = build\_lsh\_table(P, \mathbb{S}, \theta)$
    $T_Q = build\_lsh\_table(Q, \mathbb{S}, \theta)$
    while $\Delta L_{max} > 0$ do
        /* Merge meta-nodes in $P$ */
        $p_0 = random\_select(P)$
        $\Delta L_{max} = 0$, $p_{max} = undefined$
        for $p \in query\_lsh\_table(p_0, T_P)$ do
            $\Delta L = 0$
            for $q \in neighbors(p,\mathbb{S}) \cup neighbors(p_0,\mathbb{S})$ do
                $\Delta L+=$
                  $cost\_reduction\_for\_bundling((p,q),(p_0,q),R,\mathbb{S})$
            end
            $\Delta L+ = \beta_P$
            if $\Delta L > \Delta L_{max}$ then
                $\Delta L_{max} = \Delta L$, $p_{max} = p$
            end
        end
        /* Merge two clusters if cost reduction is
           possible */
        if $\Delta L_{max} > 0$ then
            $merge(p_0, p_{max}, R, \mathbb{S})$
        end
        /* Same procedure as for $Q$... */
    end
    $\theta * = \lambda_{decay}$
end
return $P$, $Q$, and $\mathbb{S}$
```

4.3 Evaluation of robustness and speed

In this section we present the experiments to evaluate our technique in terms of robustness and speed. The evaluation takes references from other similar bipartite relation summarization technique including the most recent SCMiner [11]. We also include a binary matrix reordering algorithm named Cross-association (CA) for reference [5]. We use the Python implementation for all the four algorithms. SCMiner, CA and BM-MDL-LSH use Numpy [1] which wraps C for numerical computation. We run the experiments on a Mac (OS Version High Sierra) with 2.3GHz Intel i7 CPU with 16GB RAM.

We first evaluate the robustness of our algorithm (BM-MDL-LSH) by visually inspecting the partitioning results for different bipartite graphs with ground truth co-cluster structure. The results are shown in Fig. 3. We generate three synthetic datasets in the same way as in SCMiner [11]: an empty set with no co-cluster structures Fig. 3(a), one with two one-to-one relations (Fig. 3(b)), and one relatively complex graph with more node clusters (Fig. 3(c)). For each synthetic dataset we increase the noise gradually to 10%, 30% and 50% by creating additional or missing edges in the bipartite graph. It can be observed that overall BM-MDL-LSH has a good robustness over noise. Our method creates few debris, as shown in an empty set Fig. 3(a) that when noises are increasing, the partition structure is less likely to break into more concrete pieces and the overall structure is maintained.

To evaluate the speed and the scalability the algorithm, we compare

[1]http://www.numpy.org/

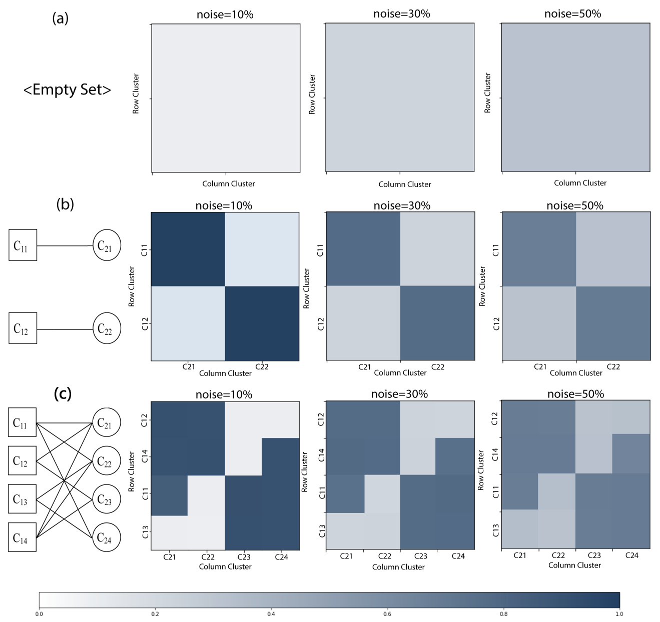


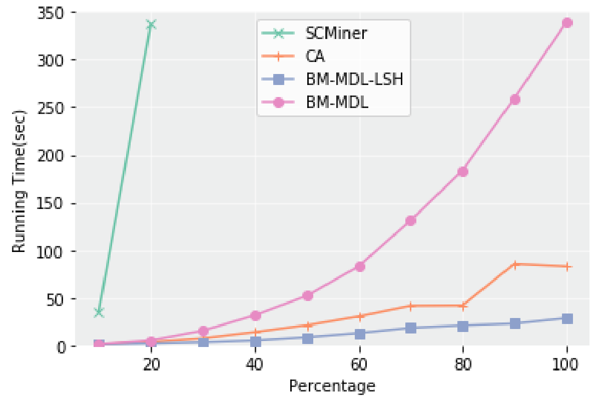
Fig. 3. Illustration of partition results using three synthetic datasets with different ground truth co-cluster structures as shown on the left of the figures. Each partition is computed three times with noise equal to 10%, 30% and 50%. The result shows that the BM-MDL-LSH algorithm is robust to noises in the data.

the running speed across SCMiner, CA, BM-MDL and BM-MDL-LSH. SCMiner and BM-MDL are similar [11, 29] since they all use 2-hop search to find candidate pairs of nodes to merge. In the experiment, we use the 1M MovieLens dataset [16]. The dataset contains 1M movie ratings from ∼6000 users on ∼4000 movies (the density is ∼4%). We compute the running speed by gradually increasing the sampling rate of the movies and the users. The results in Fig. 4 show that replacing the 2-hop search with LSH drastically reduces the running time of BM-MDL. Based on the calculation it improves the speed by more than 10 times for the full dataset. In the meanwhile, SCMiner cannot finish under reasonable time constraints — the running time exceeds 1 hour already for 50% data. Compared with the closest candidate CA, BM-MDL-LSH achieves around three times speed gain, which makes it a more practical choice for interactive bipartite relation exploration where the analyst can iteratively select different subsets in the data and run the algorithm to discover the underlying co-clusters.

Besides that we also further verify that BM-MDL-LSH does not introduce significant degradation in the results when compared to BM-MDL, in terms of description length reduction. We compare the percentage of reduction in description length in Fig. 4 for the two algorithms running on the sampled 1M MovieLens data, with the same parameter settings for α, β_P and β_Q. The result shows that BM-MDL-LSH in fact improve the results, which can be explained by the higher threshold (close to 1) we set for LSH at the initial runs that prevents dissimilar pairs of nodes from being merged. In the meanwhile, with an aggressive setting of βs (to enforce less cluster numbers), BM-MDL groups even dissimilar nodes in the initial iterations, which results in far from ideal reduction in the description length.

5 Design Requirements

While creating the visual representations and analytics system we faced many design decisions. To formulate our desiderata we interviewed a group of data scientists in the automotive industry, whose main responsibility is to analyze large amount of vehicle log data capturing the occurrences of different fault signals in cars. One typical question they are trying to answer with such log data is: are there any groups of cars that exhibit the same set of symptoms over the course of their lifetimes? Insights like this enable large scale troubleshooting and exposes hidden market segments since cars with similar faults will likely need common parts for replacement or similar services in the repair shops. We list the



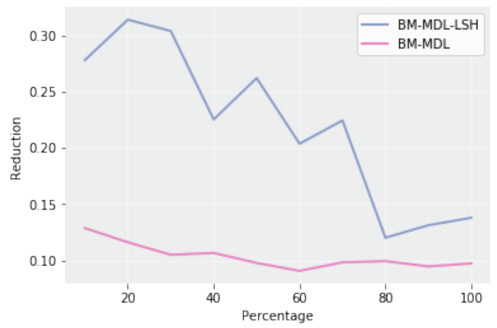


Fig. 4. Upper: comparison of running times of SCMiner, CA, BM-MDL and BM-MDL-LSH on the 1M MoiveLens dataset [16] with different sampling percentage. BM-MDL-LSH consistently outperforms others in terms of running speed. We discard all the results that require more than 1200s to complete in the plot. Lower: comparison of the description length reduction of BM-MDL and BM-MDL-LSH. It shows that BM-MDL-LSH does not degrade the basic methods and even outperforms it in most of the cases. For more detail please refer to Sect. 8.

analytic tasks gathered from the interview in the first column in Table 1.

We further investigate the possibility of generalizing the tasks in vehicle log analysis to other application domains. In Table 1 (column 2) we first generalize these tasks for an abstract bipartite graph with node attributes. Then we instantiate the analytic tasks for another real-world application: roll-call vote analysis. The instantiated tasks are also valid and critical in the corresponding application domain (political science) as verified by a domain expert[2]. Eventually we aim at designing a system that can address a core set of analytic tasks that appears in a wide range of applications applying bipartite graph to model the key relations in the data.

We categorize the tasks in Table 1 into two groups (**T.a1-4** and **T.b**). **T.a1-4** focus on the topological structure of the bipartite relation. Applying the graph summarization algorithm (BM-MDL-LSH) and visualizing the aggregated result help analysts quickly gain an overview of the data to support **T.a1-3**. However, the summary graph $\mathbb{S}$ alone is an inaccurate representation of the original data. To help analysts better assess the significance and reliability of the aggregated bipartite relations (**T.a4**), we also need to visually encode the amount of corrections $\mathbb{C}$ needed to recover the original graph. T.b focuses on the attribute values of nodes and how they are associated with the bipartite connections. The domain-specific attributes (e.g. engine type in vehicle data, gender or occupation in movie preference data, and party in roll-call votes data) add context to further understand and interpret the topological structure.

Besides supporting the analytic tasks described above, the system should also enable detail-on-demand data exploration (**R1**) since a static, high-level visual summary of data is seldom sufficient and analysts need to interactively drill down to the details for verification or identify more fine-grained structures in the data. Furthermore, the co-clustering algorithm creates node clusters and meta-edges with varying

[2]We interviewed a research scientist working in the area of political science.

Use case 1: vehicle fault analysis ($R \subseteq$ vehicles $\times$ faults)	**General case: bipartite graph analysis ($R \subseteq U \times V$)**	**Use case 2: roll-call vote analysis ($R \subseteq$ legislators $\times$ bills)**
Identify vehicles with similar faults	**T.a1** Identify nodes in U with similar bipartite connections	Identify legislators that vote similarly
Identify faults that co-occur in cars	**T.a2** Similar as T.a1 for V	Identify bills voted by similar legislators
Compare faults that occur in different vehicle clusters	**T.a3** Compare linkages between node clusters in U and V	Compare bills voted by different clusters of legislators
Assess the deviations of fault occurrence patterns for vehicles in the same cluster	**T.a4** Assess the amount of corrections needed to recover R from $\mathbb{S}$	Assess the deviations of voting patterns for legislators in the same cluster
Compare fault occurrences for cars with different shared properties e.g. engine types	**T.b** Compare bipartite connections for nodes with different attribute values	Compare voting records of different parties

Table 1. Task analysis. The two use cases correspond to the example usage scenarios are described in Sect. 7.

strengths. To help the analysts identify salient patterns in the data we should provide filtering mechanisms (**R2**) accordingly.

6 The ViBr System

We have designed ViBr to address the analytic tasks and design requirements discussed in Sect. 5, based on the summary graph generated by our technique described in Sect. 3. ViBr allows users to gain an overview of large scale bipartite relations with the summary graph (**T.a1-4**), adjust the granularity of the visualization to drill down into details (**R1**), filter the data to focus on the significant and salient cluster structures (**R2**) and apply the information encoded in domain specific node attributes for comparison, explanation and verification (**T.b**).

6.1 Visual adjacency list

To support scalable visual exploration, we design an adjacency list style visualization which is illustrated in Fig. 5 (b). The visualization represents the clusters on one side of the graph (i.e. clusters in U) as different rows and their outgoing connections to clusters on the other side (i.e. clusters in V) as colored blocks stacked from left to right. Different colors represent different node clusters in V. The height and width of the blocks are proportional to the number of nodes contained in the two clusters. Some blocks are not entirely filled to indicate that there are missing edges in the original graph. The filled height of the blocks is determined by the density of the edges. The density is the number of edges between two clusters $p \in P$ and $q \in Q$ divided by the maximum number of possible edges.

$$density(p,q) = \frac{\|p \times q \cap R\|}{\|p\| \cdot \|q\|} \tag{2}$$

$density(p,q) = 1$ if the edges between p and q form a bi-clique. The blocks are sorted from left to right based on the density of the edges connecting the two clusters of nodes.

Compared with other visualization techniques such as node-link diagram with two parallel lists of nodes (Fig. 5(a)), flow map (Fig. 5(c)) and adjacency matrix (Fig. 5(d)), the visual design results in an aligned and compact representation. It benefits the searching and understanding of bipartite relations and is adaptive to visualize graphs with different degrees of density. The key connections in the graph are prioritized and they can be easily identified by scanning vertically.

In flow map (Fig. 5 (c)) the aggregated nodes in P and Q are arranged in two parallel vertical lists and the aggregated links are drawn as curved edges connecting the corresponding nodes. The widths of the edges are modulated based on the density value computed with Equation 2. One advantage of the design is that it allows the labels to be placed horizontally which can greatly improve the readability and interpretability of the visualization. This advantage, however, diminishes when the graph becomes much larger with thousands or even more nodes. Our major concern about the flow map is the visual clutter caused by the edges crossing each other, which makes it a challenging task to gain an overview of the bipartite connections and compare subsets of data, even for graphs at a moderate scale. Recent works identify bi-cliques in the graph and bundle the edges correspondingly to reduce visual clutter [30, 39]. However these methods may fall short for large scale bipartite graphs which could contain many small

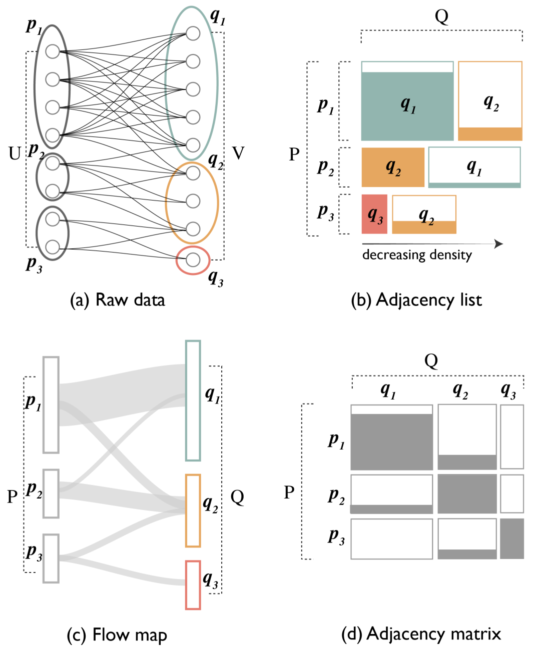


Fig. 5. Design alternatives: (a) original data with two parallel list of nodes in U and V. The node clusters identified by the algorithm are highlighted; (b) the adjacency list style design. Color encode the different node clusters in V. Each rectangle block corresponds to an aggregated edge between two clusters in U and V. The height and the width of the block are proportional to the number of the nodes in the corresponding clusters. Filled proportion in each block encodes the density of the aggregated edge. Blocked sorted from left to right by decreasing density; (c) flow map with aggregated nodes and edges; (d) adjacency matrix with aggregated rows and columns. Filled proportion in each block encodes same value as in adjacency list.

bi-cliques. Adjacency matrix (Fig. 5 (d)) is another possible design. In adjacency matrices we use the filled proportion (similar to visual adjacency list) to encode the density of the aggregated connections. Adjacency matrices are suitable for visualizing dense interconnections [14, 46]. However it lacks space efficiency: for graphs with relatively sparse interconnections the empty blocks still have to occupy the screen space and the 'data-ink' ratio is not high. Besides that, it is still a challenging task to display readable row and column labels.

One important property of the adjacency list is that by filtering the blocks with low density and small node cluster size in U and/or V we can obtain an extremely compact overview of the bipartite relations. In Fig. 6 we illustrate how the visual adjacency list is gradually simplified by increasing the threshold on density and the size of the node clusters. This property is especially useful for creating small multiples of adjacency lists to compare the bipartite relations for different subsets of data as illustrated in Fig. 7. We use animated transition in the system to further facilitate understanding.

For the default display of the aggregated graph, we choose the adjacency list style design as it is a compact design and it distinguishes

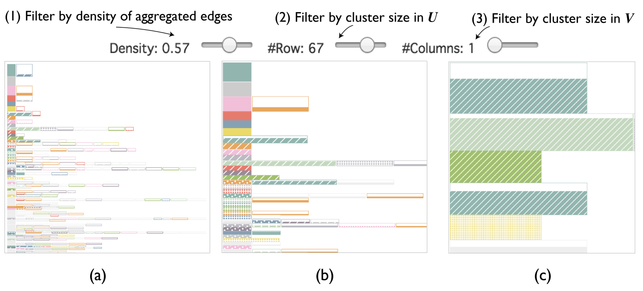

(1) Filter by density of aggregated edges
(2) Filter by cluster size in U
(3) Filter by cluster size in V
Density: 0.57
#Row: 67
#Columns: 1
(a)
(b)
(c)

Fig. 6. Filter visual adjacency list with different threshold settings on: (1)the density of the links, (2)the size of the clusters in U and (3)the size of the clusters in V. From (a) to (c) the visualization is gradually simplified and the significant patterns are highlighted.

the node clusters with vertical positions and color. Aligning rows in this way allows the separation of node clusters at the first glance. In most of the cases it looks much clearer. Color has scalability issues in assigning distinct values, but choosing it over the long horizontal axis as in adjacency matrices creates a greater utilization of space. We further extend the palette from ColorBrewer [17] with textures [43] such as ▨, ▨, ⁘. In the example usage scenarios with real-world data (Sect. 7) we realize that the distribution of bipartite connections are usually quite skewed and the color plus texture encoding can create enough variations to differentiate the major node clusters. We acknowledge that further user study is needed for a comprehensive understanding of the perceptual scalability and crafting a set of optimal designs for the texture patterns. We provide a legend (Fig. 1Ⓖ) indicating the belonging nodes to each colored cluster. The system also supports text search of the node names in the legend.

One drawback of the visual adjacency list is that the two sets of nodes (U and V) are not treated symmetrically in the visual representation. For U, the stacked heights make it easier to compare and sum the sizes of the node clusters. For V, the color encoding makes it easier to label the individual nodes in different clusters with additional legend. In practice, we also provide adjacency matrix as an alternative in the user interface and the analyst can switch between these two representations.

6.2 User Interaction

For effective exploratory data analysis, interaction is equally important as visual representation. VIBR supports the following user interactions:

Filtering: The system supports several filtering mechanisms such that analyst can focus on a particular subset or the most significant clusters in the data:

- *Filter nodes by attribute values:* VIBR supports selecting a subset of nodes based on their attribute values (Fig. 1Ⓓ) such that the analysts can focus on a relevant segment of the data.
- *Filter blocks in the adjacency list:* Although the number of node partitions can be controlled, noises in data can still produce small pieces that reduce the available spaces and affect the clarity of color encoding. Therefore, three filters are available (Fig. 6, Fig. 1Ⓑ) to remove noises based on different criteria: the density of the blocks and the corresponding size of the node clusters in U and V. Users can choose to keep only blocks that are significant in density or representative in size. The purpose is to emphasize important relations for comparison and facilitate understanding. Fig. 6(a-c) illustrates the effect of different filtering threshold settings.

Compare bipartite connections by node attributes: For comparative analysis of bipartite relations the system supports creating small multiples by slicing on a selected node attribute (Fig. 7). In the example illustrated in Fig. 7 we show the overview of the bipartite relation(Fig. 7 (a)) and the result of faceting it on a particular node attribute(Fig. 7 (b)). The result shows a unique group of nodes with very distinctive bipartite connections (Fig. 7 (c)) as the blocks have complete different color compared to the other two small multiples.

Detail-on-demand: Users can select a block to perform drill down inspection. The visual summary provides a high-level picture of the bipartite connections. However user may request more details for either verifying the results of the clustering algorithm or understanding the characteristics of a particular cluster. VIBR supports several different mechanisms for drill down inspection:

- When a mouse hovers over a block, a tooltip is displayed to show the number of rows and columns and the density of the block.
- When user clicks on a block, the detailed information of the corresponding nodes in U and V will be displayed and updated in two different tables (Fig. 1Ⓕ and Ⓖ).
- When user double clicks on a block representing the high level summary, a new window will be created to reveal low level details of the bipartite relation within it using an adjacency matrix style visualization(Fig. 1②). The co-clustering algorithm is invoked again to reorder the rows and columns in the matrix to highlight the existence of any internal structures. User can click on the matrix view then brush through the entries, so that the two data tables will be further refined to highlight the information of the selected rows and columns. Under certain circumstances improper parameter settings in BM-MDL-LSH algorithm may result in an overly aggressive compression of the data. The visualization may henceforth display nodes with drastically different bipartite connections as one single cluster. Details provided by the matrix view can help users verify the resemblance of items within the same cluster to answer their hypothesis.

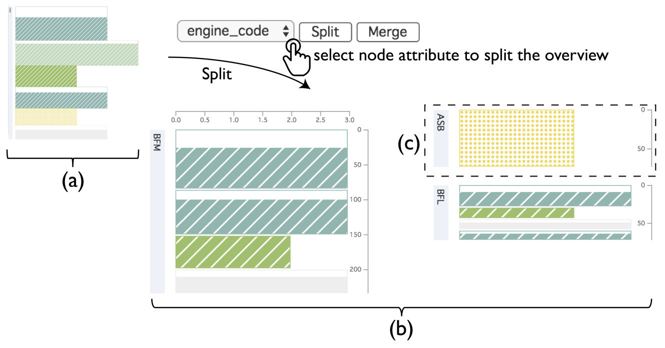


Fig. 7. The system supports creating small multiples (b) from the original visual summary (a) by slicing on a selected node attribute. This supports comparative analysis across different subsets of data. The result shows a unique group of nodes with very distinctive bipartite connections (c).

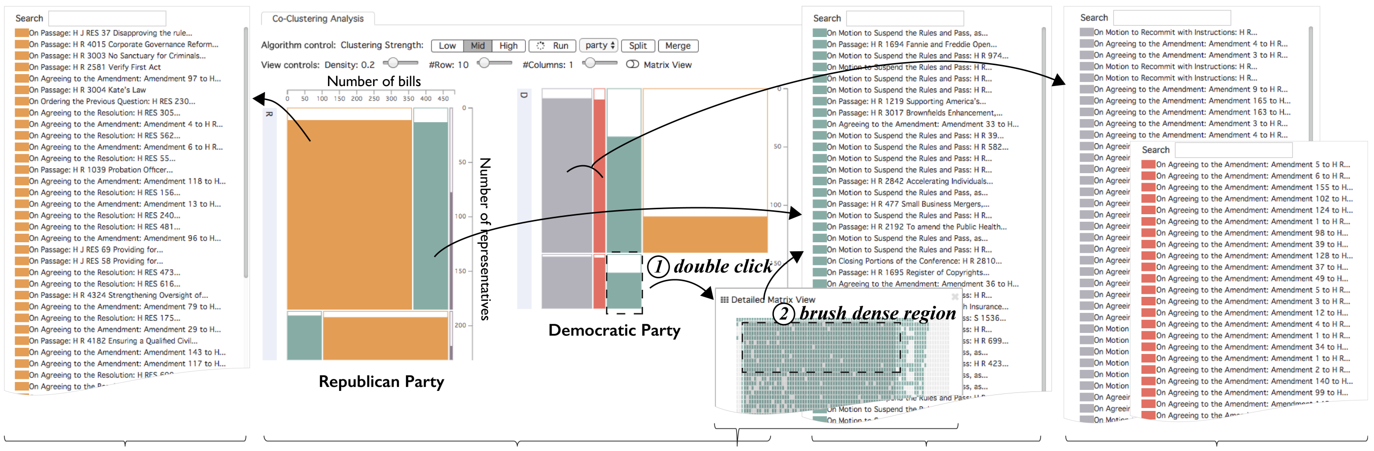
Brushing on coordinated views: The system visualizes node attribute value distributions with univariate charts: bar charts for categorical variables, histograms for (binned) numerical variables. These univariate charts are linked to the bipartite graph visualization: upon selection of a cluster the filtered data are highlighted in bar charts or histograms.

7 Example Usage Scenarios

We introduce two example usage scenarios to demonstrate the effectiveness of our techniques. First we work with a researcher in political science to apply VIBR to analyze the roll-call voting records on 668 subjects (e.g., bills, amendments, resolutions, nominations) in the 115th United States House of Representatives in 2017. The data is collected from GovTrack.us[3]. The graph is constructed based on 435 individual legislators' votes on each subject. Overall we count 170,237 favorable votes. For each favorable vote we create a bipartite connection between the corresponding legislator and the subject.

The second work consists of a group of data scientists from the automotive industry focusing on a dataset about around 7 million vehicles' after-market repair information. The dataset records the diagnostic trouble codes (DTCs) for each vehicle. We create the bipartite relations based on the occurrences of the DTCs in the individual vehicle. Each computation and visualization is bounded within an individual car model. Therefore, the analysis of each car model consists of vehicles amount ranged from 2,000 to 40,000. The exact vehicle identification number (VIN) and the DTCs are anonymized for privacy concerns.

[3]https://www.govtrack.us/


Search
Co-Clustering Analysis
Algorithm control: Clustering Strength: Low Mid High Run party Split Merge
View controls: Density: 0.2 #Row: 10 #Columns: 1 Matrix View
Number of bills
Number of representatives
Republican Party
Democratic Party
① double click
Detailed Matrix View
② brush dense region

(a) Introduction and passage of bills (b) Overview with small multiples (c) Detailed matrix view (d) Non-controversial bills (e) Recommit or amend bills

Fig. 8. Using VIBR to analyze the bipartite structure of roll-call votes in the US House of Representatives (115th Congress). (a) The Republicans mainly vote for bills that favor the legislation (orange block); (b) Overview of the bipartite relations. The two small multiples summarize the voting patterns of the Republicans and the Democrats respectively;(c) Detailed matrix view of one block; (d) Both parties vote for bills that are non-controversial. (e) The Democrats mostly vote for amendments (gray and red block). Details can be found in Sect. 7.1.

7.1 Congress voting analysis

A background of the 115th Congress is as follows: the Republicans occupied more than half of the seats in both the Senate and the House of Representatives, plus they had won the presidential election in 2016. Therefore, it is clear that they act as the ruling party while the Democrats act as the opposition party. To understand the dynamics in the congress we conducted the case study with a researcher in political science and she helped document the findings.

Overview the bipartite structure in votes. The expert first runs the co-clustering algorithm and creates a small multiple display by faceting on the party affiliation (Fig. 8(b)). The clusters of subjects being voted (e.g. bills, amendments) are assigned with different colors. The relative areas of the colored blocks in each subgraph shows that the two parties indeed have very different sets of favored bills (or amendments and etc.) and the representatives usually vote according to the consensus of their parties. Besides that, based on the total filled area of the blocks, the Republicans vote "yes" more than the Democrats, showing the proposed bills and rules are favorable to the ruling party.

Understanding roles of different parties in the legislation process. In both subgraphs we observe a teal colored block, indicating that there are quite a few number of proposals supported by both the parties. The expert clicked one teal colored block in the Democratic's partition (Fig. 8① and opened the detailed matrix view (Fig. 8(c)) for further analysis. The co-clustering algorithm runs again to reorder the rows and columns in the matrix view to reveal the internal structures in the block. The expert brushed on the matrix view to review more details (Fig. 8②) and observed that the Democrats mostly voted "yes" on the subjects about "*On Motion to Suspend the Rules and Pass*" (Fig. 8(d)), which refer to the act of quickly passing the non-controversial bills for more efficiency in the legislation process.

The orange colored subjects are rarely voted in favor by the Democrats while being supported by the majority of the Republicans as shown by its high density in the partition (Fig. 8(b). These subjects are mainly related to these categories(Fig. 8(a): 1. *Agreeing to the resolution for consideration*, which means agreeing to initiate the introduction of new bills; 2. *Ordering the previous question*, which means the motion to end debate on a pending proposal and bring it to an immediate vote; and 3. *Passage*, which means passing the proposal. It shows that as the Republican party is now the majority party, more bills and legislations of their interests and policy preferences will be proposed by them. Therefore they will mostly agree on these actions which are conducive to the establishment of new laws to fulfill their party interests. On the other hand, Democrats are more likely to oppose such initiatives and prevent the bills from passing, since those would rarely be compatible with their interests.

On the other hand, the subjects with votes mostly from the Democrats (gray and red blocks) fall into these categories (Fig. 8(e)): 1. *Motion to recommit with instructions*, which means to send back the proposal for amendment; and 2. *Agreeing to the amendment*, which refers to the amendment of several details in the bill. Our expert then comes up with an interesting question: *"Why do Democrats like to vote for amendments?"* She searches for the number of amendments voted from the Democrats, and discovers that they support amendments more than the Republicans. She then concludes that the Democrat's support on amendments is a clear reflection of the bargaining process in the legislature, as well as the general balance of power between the two parties. Amendment is a subtle issue, which may bring benefits to both the Democrats and the Republicans. As the minority party in the House, the Democrats may find the bill drafted by the Republicans far from their ideal policy position; however, this does not mean the Democrats have no bargaining power in the Congress. Instead, through negotiations and even fierce debates in the House, the Democrats may propose changes to specific articles in the drafted bill through amendments, such that their interests and preferences are represented. This helps explain why while the Democrats are reluctant to pass the bill, they are more willing to seize the opportunity for amendments.

7.2 Vehicle fault data analysis

In the second example usage scenario, we turn our attention to a completely different application domain and analyze the occurrences of faults in vehicles. The vehicle faults are recorded as DTC codes, which are warning messages generated by different electric control units (ECUs) in vehicles. They indicate abnormality in various sensor measurements or other types of malfunction in the hardware and software system. A large portion of the DTC codes are standardized across different makes and models while some of them remain unique to individual brands. Given that the vehicles are increasingly complex, the repair shops rely heavily on the historical record of DTCs to track the health statuses of the vehicles. When a vehicle visits a repair shop the mechanics will collect the on-board DTC logs and use them to perform more precise diagnosis and identify the suitable repair procedures.

Large scale analysis of DTC occurrences in vehicles help understand the demographics of vehicle faults, which would have great impact on the automotive industry. It enables large scale troubleshooting such that early warnings can be generated on the emerging fault patterns for the auto manufacturers before the problem strikes a larger vehicle population. Besides that, it also enables experience based repair by analyzing the most effective repair procedures for a particular set of co-occurring DTCs. Regardless of the size of vehicle data being analyzed, our analyst follows similar pipeline (Fig. 1) supported by VIBR to acquire insights that help understand the demographics of vehicle faults:

Search for co-clusters with high confidence. Our analyst's first action would be filtering out co-clusters by densities and sizes. In general the bipartite relations between the vehicles and the DTCs are quite sparse. Among the $\sim$3000 different types of DTCs most of them seldom occur. To seek for meaningful and significant clusters which can reflect the systematic fault patterns within a vehicle population, it is necessary to focus on clusters with a larger number of vehicles and relatively higher density. Therefore, the analyst adjusts the filters (Fig. 1 Ⓑ) available in the system such that the dense and significant blocks become more visible. (Fig. 1 Ⓐ) shows the filtered result. There are around 10 large groups of vehicles with different combinations of co-occurring DTCs. It reveals that vehicle repair and maintenance requires highly customized services and inspecting the data visually is an effective way to uncover such multi-class situation.

Compare the co-occurring faults for different clusters of vehicles. The vehicle clusters differ by either having a unique set of DTCs, or they could have a common set of DTCs but differ by some additional ones. For example, the temperature sensor problems (yellow block, Fig. 1***a***) only appear in one of the vehicle clusters, independent of all the other DTCs. While the other two groups of vehicles (Fig. 1 ***b***) share a similar set of DTCs (purple+ pink blocks) but differ by additional seat belt problems (light green block). Based on such observations the analysts can isolate different sets of vehicles for focused analysis.

Analyze the correlation of fault patterns with vehicle properties. Given the overview the analyst wants to understand whether the fault patterns are associated with a particular subpopulation of vehicles that share the same properties such as *engine type* or *country*. This can be done by partitioning the vehicles on a selected property and separating the overview into small multiples for comparison. For example, in Fig. 1 ①, each small multiple displays a summarized bipartite relation for a subset of vehicles with the same engine code. The analyst finds out that the sensor faults (orange) and pressure actuator faults (pink) only occur in vehicles with a particular type of engine (Fig. 1 ***c***).

Inspect details in the matrix view. To obtain detailed information the analysts open the matrix view. In Fig. 1 ② we give an example where a dense purple colored block is double clicked and the matrix view is displayed for the block showing the occurrences of DTCs in individual vehicles. The analyst further brushed on the corresponding entries to obtain fine-grained information of the vehicles and the DTCs such as the *VIN number* and the *body type* of the vehicles (Fig. 1 Ⓕ) and the description of the DTCs (Fig. 1 Ⓖ).

8 Expert Interview

After recording the exploratory analysis process and findings from the experts in the automotive industry and political science respectively, we gathered feedback on the effectiveness and usability of the visualizations and their thoughts on potential extensions/other applications with a few guiding questions, following the suggestions proposed in [28].

Visual design. Overall, the domain experts appreciate the clarity and novelty of the summarization as well as the visualization output. Our political science expert looks forward to not only examining votes for legislative bills, but also the exact content of it. Given the summarization now available, she can analyze the importance of certain policy issues through text analysis of the terms and phrases, which contributes to agenda setting. She believes text visualization techniques like word clouds can seamlessly be incorporated into the current system to help produce more insights regarding political issues.

Outlook to further impact. The domain experts from the automotive industry wants to add more critical labeling information in the data such as parts that are replaced in a vehicle. These can be associated with the DTC clusters identified by the algorithm, allowing repair shops to perform faster and more accurate data-driven diagnostics and even alert drivers for potential failure. Furthermore, our political science expert would like to apply the current techniques to help her research in the area of *international political economy* to understand the contemporary globalization pattern. She would like to summarize the trade between countries distributed in different geopolitical regions and compare the patterns over a period of time. She believes that it can revolutionize the traditional way of showing data within her research community.

9 Limitations and Future Work

Scalability. ***Perceptual scalability*** - The current visual design can distinguish around 40 node clusters in V without much confusion using 12 distinct colors and three different texture designs. Due to human's limited capability to distinguish colors within certain hue differences it is unlikely that the number will scale too much. It is indeed possible for some bipartite graphs to contain many fragmented co-clusters. To avoid overwhelming the analysts, one way to improve the perceptual scalability is to examine other attributes and prioritize the attention given to certain co-clusters. Besides that, we also plan to conduct *controlled user studies* to evaluate the effectiveness and the scalability of the texture design for encoding categorical variables. ***Interactive scalability*** - Although we have speed up the algorithm tremendously through LSH, it still cannot run in interactive rate (< 100ms) for large scale dataset (1M edges in Sect. 4.3). We plan to explore more strategies to further speed up the algorithm.

Weighted bipartite relations. Our algorithm only consider binary bipartite relations. Many real-world data contain weighted bipartite relations: customers purchase the same items for multiple times in sales records, words or phrases appear multiple times in a document, users give a range of ratings (e.g. from one to five) on books and movies instead of binary likes and dislikes and genes have different level of expressions in bodies. In such cases simply thresholding the values to create a binary bipartite graph may not be a good option. We will continue to explore the possibility of incorporating numerical regularities to compress weighted bipartite relations and adapt the visualization and interaction techniques for weighted co-clusters.

Extensibility to multi-mode graphs. Our algorithm and visualization design currently only focus on summarizing bipartite relation between two sets. In many real-world applications there are more than two types of interconnected entities. For example, bibliography data contain several different set of entities including authors, papers, keywords and journal/conferences. The vehicle fault data may contain entities including faults, vehicles and suppliers of the corresponding parts. Designing a meaningful two-part representation for such multi-mode relations and applying the MDL principle to extract the key patterns that span across multiple dimensions would be a promising research direction to explore in the future.

Detect overlapping co-clusters. Our problem formulation currently supports identifying disjoint node clusters. However, it could be imagined that in some application scenarios overlapping node clusters could be quite meaningful, e.g. researchers belonging to several different communities. Addressing these application scenarios requires changing the formulation of the two-part representation, the optimization algorithm, as well as the visual designs to support a different set of user tasks.

Application domains. There are a myriad of real life examples and applications in which we can harness the power of such techniques to conduct more sophisticated human-centered analysis. For example, transaction data in market basket analysis, which uses association rules mining to acquire insights, can be benefited by interactive visualization of summarizing the bipartite structure between the transaction logs and the generated frequent item-sets.

10 Conclusion

In this paper, we visit the problem of summarizing large scale bipartite relations and introduce a novel interactive visual analytics approach to address the challenges. First, we propose an information-theoretic co-clustering algorithm based on the MDL principle. The algorithm runs efficiently on large scale bipartite graphs which makes it suitable to support interactive visual exploration. After that, we present a comprehensive visual analytics system with a novel visual adjacency list style design and multiple levels-of-detail to facilitate interactive exploration of large scale bipartite graphs with multivariate node attributes. We present example usage scenarios with two real-world dataset and collect feedback from the domain experts to demonstrate the effectiveness of the proposed approach.

References

[1] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. Visualizing sets and set-typed data: State-of-the-art and future challenges. In *Eurographics conference on Visualization (EuroVis)–State of The Art Reports*, pp. 1–21, 2014.

[2] J. Bertin. Semiology of graphics: diagrams, networks, maps. 1983.

[3] N. Cao, J. Sun, Y.-R. Lin, D. Gotz, S. Liu, and H. Qu. Facetatlas: Multifaceted visualization for rich text corpora. *IEEE transactions on visualization and computer graphics*, 16(6):1172–1181, 2010.

[4] C. Carrubba, M. Gabel, and S. Hug. Legislative voting behavior, seen and unseen: A theory of roll-call vote selection. *Legislative Studies Quarterly*, 33(4):543–572, 2008.

[5] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 79–88. ACM, 2004.

[6] Y.-Y. Chan, F. Chirigati, H. Doraiswamy, C. T. Silva, and J. Freire. Querying and exploring polygamous relationships in urban spatio-temporal data sets. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1643–1646. ACM, 2017.

[7] Y. Chen, P. Xu, and L. Ren. Sequence synopsis: Optimize visual summary of temporal event data. *IEEE transactions on visualization and computer graphics*, 2017.

[8] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire. Data polygamy: the many-many relationships among urban spatio-temporal data sets. In *Proceedings of the 2016 International Conference on Management of Data*, pp. 1011–1025. ACM, 2016.

[9] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 269–274. ACM, 2001.

[10] M. Dörk, N. H. Riche, G. Ramos, and S. Dumais. Pivotpaths: Strolling through faceted information spaces. *IEEE transactions on visualization and computer graphics*, 18(12):2709–2718, 2012.

[11] J. Feng, X. He, B. Konte, C. Böhm, and C. Plant. Summarization-based mining bipartite graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1249–1257. ACM, 2012.

[12] P. Fiaux, M. Sun, L. Bradel, C. North, N. Ramakrishnan, and A. Endert. Bixplorer: Visual analytics with biclusters. *Computer*, 46(8):90–94, 2013.

[13] S. Ghani, B. C. Kwon, S. Lee, J. S. Yi, and N. Elmqvist. Visual analytics for multimodal social network analysis: A design study with social scientists. *IEEE transactions on visualization and computer graphics*, 19(12):2032–2041, 2013.

[14] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pp. 17–24. Ieee, 2004.

[15] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pp. 512–521. IEEE, 1999.

[16] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

[17] M. Harrower and C. A. Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[18] M. Hlawatsch, M. Burch, and D. Weiskopf. Visual adjacency lists for dynamic graphs. *IEEE transactions on visualization and computer graphics*, 20(11):1590–1603, 2014.

[19] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[20] B. Kim, B. Lee, and J. Seo. Visualizing set concordance with permutation matrices and fan diagrams. *Interacting with computers*, 19(5-6):630–643, 2007.

[21] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003.

[22] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[23] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2017.

[24] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, 2014.

[25] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. *arXiv preprint arXiv:1710.10777*, 2017.

[26] K. Misue. Drawing bipartite graphs as anchored maps. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60*, APVis '06, pp. 169–177. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2006.

[27] K. Misue and Q. Zhou. Drawing semi-bipartite graphs in anchor+ matrix style. In *Information Visualisation (IV), 2011 15th International Conference on*, pp. 26–31. IEEE, 2011.

[28] T. Munzner. A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, 15(6), 2009.

[29] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 419–432. ACM, 2008.

[30] Y. Onoue, N. Kukimoto, N. Sakamoto, and K. Koyamada. Minimizing the number of edges via edge concentration in dense layered graphs. *IEEE transactions on visualization and computer graphics*, 22(6):1652–1661, 2016.

[31] C. Perin, P. Dragicevic, and J.-D. Fekete. Revisiting bertin matrices: New interactions for crafting tabular visualizations. *IEEE transactions on visualization and computer graphics*, 20(12):2082–2091, 2014.

[32] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[33] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE transactions on visualization and computer graphics*, 12(6):1427–1439, 2006.

[34] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE transactions on visualization and computer graphics*, 12(5):733–740, 2006.

[35] H. Siirtola and E. Mäkinen. Constructing and reconstructing the reorderable matrix. *Information Visualization*, 4(1):32–48, 2005.

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

[37] M. Streit, S. Gratzl, M. Gillhofer, A. Mayr, A. Mitterecker, and S. Hochreiter. Furby: fuzzy force-directed bicluster visualization. *BMC bioinformatics*, 15(Suppl 6):S4, 2014.

[38] M. Sun, L. Bradel, C. L. North, and N. Ramakrishnan. The role of interactive biclusters in sensemaking. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pp. 1559–1562. ACM, 2014.

[39] M. Sun, P. Mi, C. North, and N. Ramakrishnan. Biset: Semantic edge bundling with biclusters for sensemaking. *IEEE transactions on visualization and computer graphics*, 22(1):310–319, 2016.

[40] M. Sun, C. North, and N. Ramakrishnan. A five-level design framework for bicluster visualizations. *IEEE transactions on visualization and computer graphics*, 20(12):1713–1722, 2014.

[41] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *International Conference on Discovery Science*, pp. 16–31. Springer, 2004.

[42] R. Veras and C. Collins. Optimizing hierarchical visualizations with the minimum description length principle. *IEEE transactions on visualization and computer graphics*, 23(1):631–640, 2017.

[43] C. Ware. *Information visualization: perception for design*. Elsevier, 2012.

[44] P. Xu, N. Cao, H. Qu, and J. Stasko. Interactive visual co-cluster analysis of bipartite graphs. In *Pacific Visualization Symposium (PacificVis), 2016 IEEE*, pp. 32–39. IEEE, 2016.

[45] M. J. Zaki and C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE transactions on knowledge and data engineering*, 17(4):462–478, 2005.

[46] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. Matrixwave: Visual comparison of event sequence data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 259–268. ACM, New York, NY, USA, 2015. doi: 10.1145/2702123.2702419

[47] J. Zhao, M. Sun, F. Chen, and P. Chiu. Bidots: Visual exploration of weighted biclusters. *IEEE transactions on visualization and computer graphics*, 2017.