# Interactive Bicluster Aggregation in Bipartite Graphs

Maoyuan Sun* David Koop* Jian Zhao† Chris North‡ Naren Ramakrishnan‡
*Northern Illinois University †University of Waterloo ‡Virginia Tech

## Abstract

Exploring coordinated relationships is important for sensemaking of data in various fields, such as intelligence analysis. To support such investigations, visual analysis tools use biclustering to mine relationships in bipartite graphs and visualize the resulting biclusters with standard graph visualization techniques. Due to overlaps among biclusters, such visualizations can be cluttered (e.g., with many edge crossings), when there are a large number of biclusters. Prior work attempted to resolve this problem by automatically ordering nodes in a bipartite graph. However, visual clutter is still a serious problem, since the number of displayed biclusters remains unchanged. We propose bicluster aggregation as an alternative approach, and have developed two methods of interactively merging biclusters. These interactive bicluster aggregations help organize similar biclusters and reduce the number of displayed biclusters. Initial expert feedback indicates potential usefulness of these techniques in practice.

**Keywords:** Bicluster, bipartite graph, visual analytics.

## 1 Introduction

Biclustering has been employed in visual analysis tools to support investigating coordinated relationships [5, 13, 15, 18]. Coordinated relationships are groups of shared relations between sets of entities, and are involved in analytical tasks in various fields. For example, intelligence analysts explore coordinated activities (e.g., six people visited the same five cities on the same four days) from intelligence reports to identify potential threats [8]. Security analysts investigate coordinated communications among applications to detect malware collusions [11]. Bioinformaticians examine coordinated relations between genes and conditions to find similar genes [1].

A bicluster is a relationship between two sets of entities such that each entity in one set is related to all in the other. Thus, it can reveal a particular coordinated relationship. Moreover, biclusters can be algorithmically mined from a dataset [10]. However, computed biclusters can overlap with each other by sharing certain entities. Edge bundles between entity lists can be used to show each bicluster, and the resulting visualization based on bipartite graphs has been proposed and studied [16].

When the number of computed biclusters is large, such visualization can get cluttered easily. Prior work attempted to resolve this by manipulating the orders of entities and bundles in the visualization [20]. While ordering can help to reduce edge crossings, the number of biclusters remains untouched. Thus, the visual clutter issue may not be well addressed all the time, since the challenge of reasonably reducing the number of presented biclusters endures.

To tackle this challenging problem, we propose two ways of interactively aggregating biclusters in this bipartite graph-based visualization. Automatic aggregation supports users merging groups of biclusters at once, while manual aggregation shows similar biclusters to users in detail, and allows users to pick several of them for aggregation. They help to reduce the number of displayed biclusters and organize similar biclusters together, thus facilitating users with visual analysis of coordinated relationships. Moreover, we discuss our design rationale for the two aggregation techniques, and present initial expert feedback about them.

*e-mail: {smaoyuan, dakoop}@niu.edu. Work was completed while the authors were at the University of Massachusetts Dartmouth.

†e-mail: jianzhao@uwaterloo.ca. Work was completed while the author was at FXPAL.

‡e-mail: {north, naren}@cs.vt.edu

## 2 Background

### 2.1 Bicluster

Biclusters are computational results from biclustering algorithms, which attempt to find both subsets of entities and dimensions such that for each identified subset of entities, they have identical behaviors within the corresponding subset of dimensions [10]. A bicluster reveals a coordinated relationship between two entity sets, and an entity set is a set of unique elements from a specific domain (e.g., people) that are extracted from a dataset (e.g., documents).

Our notion of biclusters, in this paper, follows that discussed in prior work [19]. A *bicluster* $(A', B')$ on $R\ (A, B)$ is defined as a set $A' \subseteq A$ and a set $B' \subseteq B$ such that $A' \times B' \subseteq R$, where $R\ (A, B)$ denotes a relationship between $A$ and $B$, which is a subset of $A \times B$ (the Cartesian product of $A$ and $B$). That is, every element in $A'$ is related with each element in $B'$. In different scenarios, there are different ways to model the relationship $R$. For example, $R$ can be determined by word co-occurrence in text analytics, or based on communications between web applications in cyber security. Also, the total number of entities denotes the **size** of a bicluster.

Computed biclusters can overlap each other by sharing some entities, so a bicluster can be similar to another, to some extent. *Jaccard index* (a.k.a., *Jaccard similarity coefficient*) is a useful measure to compute set similarity, and it has been used to determine the similarity between two biclusters [7].

### 2.2 Visualizing Biclusters in Bipartite Graphs

Considering components of a bicluster, there are three design strategies to show biclusters: *entity*-driven, *relationship*-driven and *cluster*-driven [23], respectively focusing on entities, relationships between entities, and a bicluster as a whole. An entity-driven design, without duplicating entities, applies a certain set visualization technique (e.g., Line Sets [2] or Bubble Sets [4]) to reveal biclusters. A relationship-driven design emphasizes relationships over entities. It often uses a matrix to present relationships, and requires that users interactively order the matrix to view different biclusters (e.g., Bicluster Viewer [6]). A cluster-driven design highlights each bicluster with its own visual mark, which allows duplicate entities and relationships (e.g., Bixplorer [5], Furby [15], and BiDots [23]).

Due to overlap, it is not always easy to clearly show biclusters by relying on their components without any duplication. This has been identified as a key design challenge for bicluster visualization, and led to BiSet, a bipartite graph-based visualization technique [18]. In BiSet, entities are organized in lists, and their relationships are shown as edges that connect entities from different lists. Each bicluster is displayed as an edge bundle between two entity lists, which merges relations of a bicluster and forms bicluster lists. This design attempts to combine the three strategies: entities, relationships, and biclusters, each having their own visual mark. While BiSet allows users to order

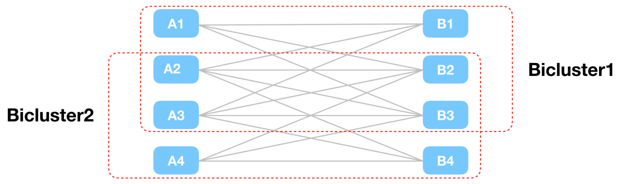

Figure 1: Two biclusters overlap by sharing entities: A2, A3, B2, B3.

entities and biclusters, the visualization can get cluttered easily, since ordering does not change the number of displayed biclusters.

## 3 Design Rationale

While this edge bundle based approach is flexible for users to organize and manipulate both entities and biclusters, it still gets cluttered due to the number of displayed biclusters, entities, or relationships. Given a fixed display area, it is hard to resolve the clutter without changing the number of visual elements presented. For example, hundreds of biclusters can be mined from the Yeast and Human B-cell Lymphoma datasets [3]. Our key goal is to reduce the information displayed. In this work, as a starting point, we develop aggregation techniques with a relatively small number (e.g., dozens of biclusters in a bipartite graph).

It is not reasonable to remove some biclusters randomly, since this may result in a loss of useful information for analysis. Instead, we propose to use similarity between biclusters as a measure of the amount of overlap and thus the likelihood that users might wish to merge them. By aggregating similar biclusters, it is possible to reduce the number of displayed visual elements. Our goal, then, is to enable users to locate and aggregate similar biclusters.

At the same time, each bicluster has its unique information (e.g., entities). For example, as is shown in Figure 1, besides sharing four entities, the two biclusters have their own unique entities. When aggregating information from biclusters, we may form certain new "bicluster", a structure consisting of relations between entities from two sets, such as $\{A1, A2, A3, A4\}$ and $\{B1, B2, B3, B4\}$. The new structure is similar to a bicluster, but not all entities in one set are related to all entities in another (e.g., no connection between $A1$ and $B4$). We call this type of structure, a **merged bicluster**, although it does not fully follow the definition of bicluster (see Section 2.1). Prior studies found that users can intentionally aggregate biclusters like this, and use them for analyses, even when such aggregation is not visualized [17, 20]. Also, for merged biclusters, users need to know both shared and unique information. Thus, it is important to show users both aspects for a merged bicluster.

In summary, our design goals in this work are: **(G1)** facilitating users with finding similar biclusters; **(G2)** supporting users in merging biclusters based on similarity; and **(G3)** visualizing merged biclusters with both shared and unique information.

## 4 Interactive Bicluster Aggregation

Driven by these goals, we propose two interactive ways of bicluster aggregation and build them on top of BiSet. It organizes entities and biclusters in different lists, so users can easily see them. Compared with matrix-based representations, BiSet does not duplicate entities to display biclusters, even for those overlapping. When performing bicluster aggregation, users can focus on bicluster-lists.

### 4.1 Bicluster Similarity Computation

Each bicluster includes two sets of entities that come from different domains (e.g., student and class). Based on which set(s) a user may emphasize on (e.g., student only, class only, or both), there are three possible ways of computing similarity between two biclusters. We design the following weighted Jaccard index to measure the similarity between two biclusters, $a$ and $b$ (G1). It aims to enable flexible user adjustment to emphasize one set over the other.

$$J_w(a,b) = w \cdot J_c(a,b) + (1-w) \cdot J_r(a,b), \quad (0 \le w \le 1) \qquad (1)$$

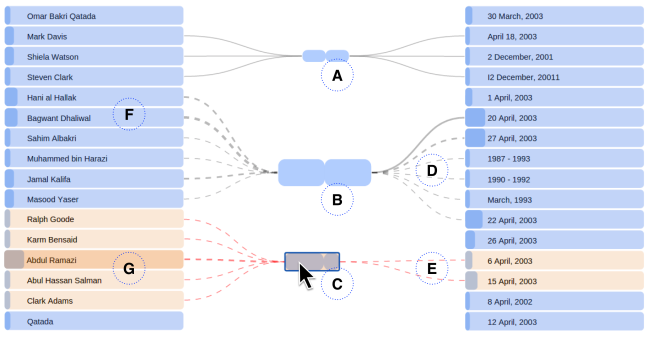

Figure 2: An example of visual encodings. (A), (B) and (C) present a bicluster, a merged bicluster, a merged bicluster (under selection), respectively. (D) and (E) show edges connecting merged biclusters with related entities, and those in (E) are highlighted. (F) and (G) are entities in the normal state and those in the highlighting state.

$J_c(a,b)$ indicates the similarity between the two biclusters on one entity set (e.g., student), and $J_r(a,b)$ reveals the similarity between them on another entity set (e.g., class). $J(\cdot)$ is the Jaccard index. Moreover, $w$ is a user selected weight that reveals how much one entity set is emphasized.

Based on Equation (1), to support interactive aggregation, we add two sliders, a **weight** slider and a **threshold** slider, to each bicluster-list in BiSet (G2). The former enables users to control the weight used in the equation. For instance, the more a user slides it to the right, the more emphasis is given to the similarity between the set 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, when the computed similarity between biclusters is greater than the value selected 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 support both **automatic** and **manual** bicluster aggregation, and they share the same visual encodings of merged biclusters.

### 4.2 Visual Encoding

Figure 2 gives an example of visual encodings that support bicluster aggregation (G3). There are two entity-lists and one bicluster-list. For entities, we keep the same visual encodings as those used in BiSet. Each entity is displayed as a blue rectangle and a small rectangle on the left of an entity shows its frequency in a dataset (Figure 2F). Biclusters are displayed as two horizontally adjacent rectangles with rounded corners. The length of these rectangles reveals the number of related entities (on the left or right), and the height of them are identical for biclusters (e.g., Figure 2A), which do not include merged ones. For a merged bicluster, the height of the two rectangles is determined by the number of biclusters that are aggregated. The more biclusters are aggregated, the higher these two rectangles are. For example, for the merged biclusters B and C in Figure 2, B aggregates more biclusters than C.

Moreover, a merged bicluster has two types of edges, connecting itself with its related entities, visually displayed as *normal* curves and *dashed* curves (Figure 2D). The former indicates that all biclusters, involved in an aggregation, share an entity (e.g., *20 April, 2003*), while the latter reveals that not all biclusters in an aggregation are related to an entity. The width of dashed curves is determined by the number of biclusters that share an entity. For example, in Figure 2, for the merged bicluster B, the number of its biclusters related with *B. Dhaliwal*, is larger than those sharing *S. Albakri*. When a user selects a merged bicluster, its related entities are highlighted. In concert with dashed curves, entities shared by more biclusters are highlighted with a brighter color (Figure 2G). With these visual encodings, a merged bicluster and its associated edges help to reveal the information of biclusters that are aggregated.

**Algorithm 1:** Similarity based aggregation in a bicluster-list

```
input  : $bics$, a set of biclusters
         $jMatrix$, a matrix of pairwise similarity of biclusters
         $thresVal$, a threshold to control bicluster aggregation
output : $bicSets$, a set of sets of biclusters to be merged
1  while $bics$ is not empty do
2      $aSet$ = new Set();
3      $aSet$.add($bics$.pop());
4      foreach $bic$ in $bics$ do
5          $movingFlag = 0$;
6          foreach $member$ in $aSet$ do
7              if $jMatrix[member][bic] \geq thresVal$ then
8                  $movingFlag$ += 1;
9          if $movingFlag = aSet.length$ then
10             $aSet$.add($bic$);
11     $bicSets$.add($aSet$);
12     while $aSet$ is not empty do
13         $bics$.remove($aSet$.pop());
14 return $bicSets$;
```

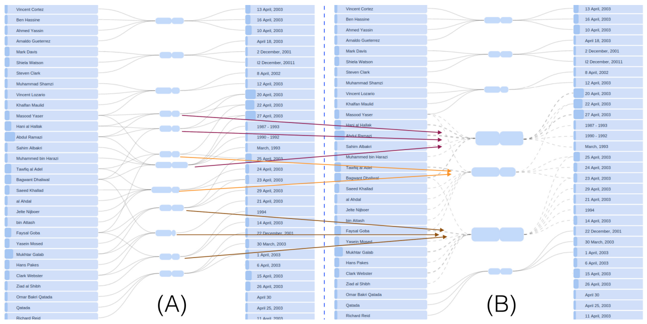

Figure 3: An example of automatic aggregation. (A) shows the original layout. (B) is the result of applying aggregation to the layout in (A).

### 4.3 Automatic Aggregation

Automatic aggregation allows users to quickly aggregate biclusters and get multiple merged biclusters at once. It is executed after a user adjusts the value in either of the two sliders (G2), and aggregation results are directly displayed without animation. Merged biclusters are determined by similarity that is computed using Equation (1) (G1). We use a greedy approach for automatic bicluster aggregation by partitioning a bicluster-list into groups of biclusters (Algorithm 1) (G2). Based on the partition, biclusters within the same group are further aggregated together. Automatic aggregation helps to reduce information displayed on the screen, including both biclusters and edges. Figure 3 shows an example with weight and threshold set as *0.1*. Comparing Figure 3A with Figure 3B, the number of bicluster reduces about 41% (from 12 to 7), and the number of edge decreases around 14% (from 71 to 61).

### 4.4 Manual Aggregation

Manual aggregation allows users to specify similar biclusters for merging. To support manual aggregation, we enhance BiSet with two interactive features (G2): **attraction** and **similarity radar**. Figure 4 shows an example of using the two features for bicluster aggregation. Figure 4A and Figure 4B 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., separated groups of biclusters in Figure 4B). Figure 4C shows 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 Figure 4C and Figure 4A). Moreover, by interacting with the similarity radar, users can specify and merge several biclusters together (Figure 4D and Figure 4E).

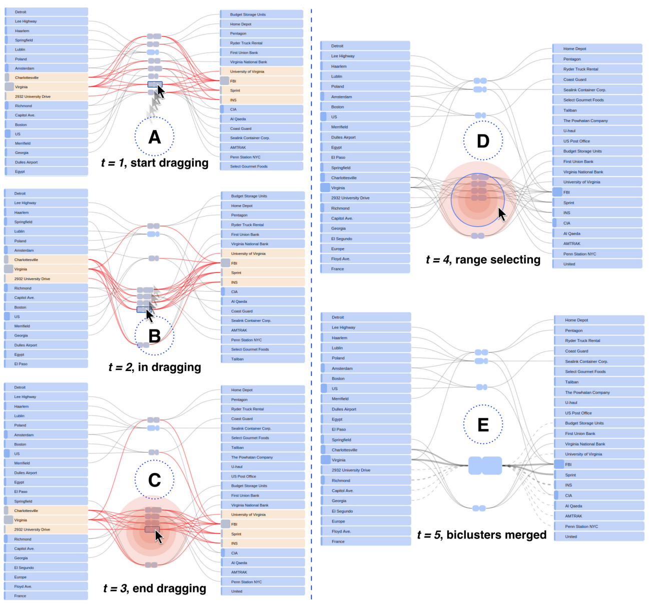

Figure 4: An example of manual bicluster aggregation by using two interactive features: attraction and similarity radar. (A), (B) and (C) sequentially shows the attraction as users drag a bicluster. (D) and (E) shows manual bicluster aggregation by using a similarity radar.

#### 4.4.1 Attraction

Attraction reveals similar biclusters and enables users to organize them as well as related entities based on the one being investigated. We apply the *dust and magnet* visual metaphor [22] at two levels (*bicluster*-level and *entity*-level) to support attraction (G2). As shown in Figure 4, 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 automatic aggregation, the similarity calculation is based on user-selected values using the two sliders, discussed before. Moreover, by right-clicking menus on biclusters, users can choose to enable or disable attraction (when they drag a bicluster). After attraction is enabled, when dragging a bicluster, the distance between each similar bicluster to the user-dragged one is determined based on the similarity with a linear mapping function (G2). The more similar a bicluster is, the shorter the distance it is from the dragged one.

#### 4.4.2 Similarity Radar

The similarity radar plays two roles: **indicating similarity levels**, and **offering handles for manual aggregation**. A similarity radar includes a group of concentric ring areas in red (with different brightness to reveal similarity levels), and the user dragged bicluster locates in its center (Figure 5). The number of ring areas is determined by the number of unique, computed similarity values, and biclusters, with different similarity values, are placed in different ring areas (G2). 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 center one. The more inner ring area (visually with more saturated red) a bicluster locates, the more similar it is to the center one. For example, in Figure 5, bicluster D is more similar to bicluster A than bicluster E is.

Moreover, a similarity radar also helps to reveal the size difference among similar biclusters (G2). If the size of a similar bicluster is larger (or smaller) than the user dragged one, it locates above (or below) the center one. For example, in Figure 5, the similarity value between biclusters C and A is the same with that between biclusters D and A, as biclusters C and D are placed in the same ring area. Bicluster C has a bigger size than bicluster A, while the size of bicluster D is smaller than bicluster A. Thus, bicluster C is placed above bicluster A, in Figure 5, but bicluster D is below bicluster A.

In some cases, multiple biclusters share the same similarity value. If this happens, they are placed in the same ring area, so the size of a particular ring area is enlarged. This helps to visually highlight a group of biclusters that are equally similar to a user dragged one. In addition, within such a ring area, similar biclusters are organized by size, with the strategy discussed before.

Similar to the popup widget for subtree expanding or collapsing in [12], a similarity radar enables users to merge a group of similar biclusters by clicking on ring areas (G2). When a user hovers the mouse over a ring area, its boundary (a circle) is highlighted (see Figure 4D). After users click on a ring area, biclusters within the highlighted boundary are merged (Figure 4E). These ring areas work as handles to support users manually merging a group of similar biclusters. Compared with manually selecting individual biclusters for aggregation, such a similarity radar based bicluster aggregation 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.

## 5 Initial Expert Feedback

To evaluate the proposed interactive techniques, we conducted an interview with a marketing analyst at an IT company with ten years of experience. His job requires performing analyses on transaction data to understand market trends and opportunities. Due to the large amount of transactions (e.g., millions of records) in the real world, he often starts with small samples (e.g., a few hundred or thousand records) to try several analysis methods, and then applies them to the whole dataset. He indicated the potential usefulness of interactive bicluster aggregation in practice and also suggested improvements.

In the interview, he explored biclusters in a bipartite graph of customer and product feature usage, extracted from a company's sales data. First, he appreciated the function of bicluster aggregation: "*Merging biclusters is helpful for me to find a bigger group of customers, or set some new collections of features for sales. Also, it makes my view looks better [less cluttered].*" Considering the two ways of bicluster aggregation, he thought automatic aggregation was more efficient: "*My job usually play with big data, so automatically merging several ones works better for me.*" He also confirmed the usefulness of manual aggregation: "*If I had time, I would definitely like to check and merge information from biclusters by myself. I like a lot the map [similarity radar] that encloses similar biclusters, as it allows me to make a decision [of aggregation].*"

Besides the positive comments, he raised three suggestions for future improvement. First, regarding to similarity calculation, his comment indicated that Jaccard index-based approach was not enough: "*These five features [in one bicluster] and those three features [in another bicluster] should be grouped together, because they were features about data integration service, but the tool did not find that.*" Second, he suggested more flexible ways of aggregation: "*Sometimes I just want some parts [of information] from biclusters, not merging all [information from biclusters]. It would be great, if I could do this, or make some changes to a merged bicluster.*" Third, he requested the ability to separating merged biclusters: "*Can I break this [merged] bicluster apart?*"

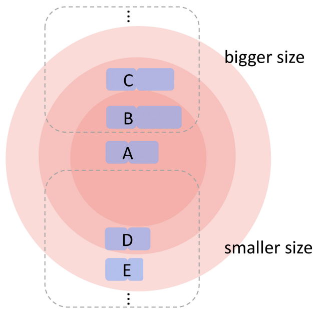

Figure 5: An example of a similarity radar. (A) shows a user dragged bicluster. (B), (C), (D) and (E) are similar biclusters.

## 6 Conclusion and Future Work

We have presented two interactive approaches to aggregate biclusters. Automatic aggregation supports users quickly merging groups of biclusters. Manual aggregation shows similar biclusters and allows users to select several of them for aggregation. By merging several groups of biclusters at once, automatic aggregation both reduces visual clutter and helps users gain a quick overview. Compared to the former, manual aggregation needs more user-steering. It pulls, shows, and aggregates related bicluster based on a user specified one. They can be complementary to each other by supporting visual analysis with either an "overview first" [14] strategy or an "analyze first" [9] strategy.

Initial expert feedback shows potential usefulness of our proposed aggregation techniques in practice. While this work demonstrates the feasibility of interactive bicluster aggregation in bipartite graphs, it has three limitations that need further studies.

First, using dashed curves for a merged bicluster may not scale well. It reveals that an entity is not shared by all biclusters already merged. However, if many dashed curves are displayed, due to a large number of partially shared entities involved in aggregation, it will cause visual clutter, and visually tracing dashed curves to check connections is not easy. Also, it may not be effective by using the width of a dashed curve to indicate the number of biclusters sharing an entity. Moreover, the drag-and-drop based interaction for manual aggregation cannot work well, when there are too many biclusters. For example, it is hard for users to drag a bicluster that is visually covered by a few others. To address them, we plan to study other visual encodings and interaction-supportive features, including fading parts of dashed curves to reduce visual clutter, clustering biclusters, and zoom in/out the area displaying biclusters.

Second, this work currently only supports bicluster aggregation. Based on shared entities, biclusters can be linked together to form bicluster-chains [19], for a n-partite dataset. While this work can only merge a specific part of chains, it is possible to extend the technique from bicluster-level to chain-level. To compute similarity between chains, we can consider each chain as a set of multiple sets of entities and extend Equation (1) by considering more sets. Moreover, Hao et al. have studied the surprisingness of related chains for a given bicluster [21], so it is also possible to use the computed surprisingness score to merge chains. To enable chain-level aggregation, we plan to add extra visual encoding (e.g., a colored ribbon) to show a specific chain, further propagate the dust and magnet metaphor to multiple lists, and add UI widgets for users to control chain-level aggregation as parts of chains or entire chains.

Third, the initial evaluation of our proposed techniques came from one domain expert only. In order to gain an in-depth understanding of the impact of these techniques on visually exploring coordinated relationships, we need to conduct user studies in future, using both synthetic and real world datasets. One key research question that we plan to answer is: what are the trade-offs when using bicluster aggregation? Based on our prior user studies on biclusters [5,17,20], it takes users some time (e.g., from one to several minutes) to find and understand a bicluster. Thus, instead of using simple, benchmark tasks (e.g., asking users to merge three most relevant biclusters from a list of biclusters), our future study will focus on users' strategies of aggregating biclusters and using them.

## Acknowledgments

This work was supported in part by a NSF grant, IIS-1850036.

## References

[1] D. B. Allison, X. Cui, G. P. Page, and M. Sabripour. Microarray data analysis: from disarray to consolidation and consensus. *Nature reviews genetics*, 7(1):55, 2006. doi: 10.1038/nrg1749

[2] B. Alper, N. Riche, G. Ramos, and M. Czerwinski. Design study of linesets, a novel set visualization technique. *IEEE transactions on visualization and computer graphics*, 17(12):2259–2267, 2011. doi: 10.1109/TVCG.2011.186

[3] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the 2004 SIAM international conference on data mining*, pp. 114–125. SIAM, 2004. doi: 10.1137/1.9781611972740.11

[4] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009. doi: 10.1109/TVCG.2009.122

[5] P. Fiaux, M. Sun, L. Bradel, C. North, N. Ramakrishnan, and A. Endert. Bixplorer: Visual analytics with biclusters. *Computer*, 46(8):90–94, 2013. doi: 10.1109/MC.2013.269

[6] J. Heinrich, R. Seifert, M. Burch, and D. Weiskopf. Bicluster viewer: a visualization tool for analyzing gene expression data. In *International Symposium on Visual Computing*, pp. 641–652. Springer, 2011. doi: 10.1007/978-3-642-24028-7_59

[7] S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. Van Sanden, et al. Fabia: factor analysis for bicluster acquisition. *Bioinformatics*, 26(12):1520–1527, 2010. doi: 10.1093/bioinformatics/btq227

[8] F. Hughes and D. Schum. Discovery-proof-choice, the art and science of the process of intelligence analysis-preparing for the future of intelligence analysis. *Washington, DC: Joint Military Intelligence College*, 2003.

[9] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. In *Visual data mining*, pp. 76–90. Springer, 2008. doi: 10.1007/978-3-540-71080-6_6

[10] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1):24–45, 2004. doi: 10.1109/TCBB.2004.2

[11] C. Marforio, H. Ritzdorf, A. Francillon, and S. Capkun. Analysis of the communication between colluding applications on modern smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 51–60. ACM, 2012. doi: 10.1145/2420950.2420958

[12] M. J. McGuffin and R. Balakrishnan. Interactive visualization of genealogical graphs. In *IEEE Symposium on Information Visualization*, pp. 16–23. IEEE, 2005. doi: 10.1109/INFVIS.2005.1532124

[13] R. Santamaría, R. Therón, and L. Quintales. Bicoverlapper 2.0: visual analysis for gene expression. *Bioinformatics*, p. btu120, 2014. doi: 10.1093/bioinformatics/btu120

[14] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pp. 336–343. IEEE, 1996. doi: 10.1109/VL.1996.545307

[15] 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. doi: 10.1186/1471-2105-15-S6-S4

[16] M. Sun. *Visual Analytics with Biclusters: Exploring Coordinated Relationships in Context*. PhD thesis, Virginia Tech, 2016.

[17] M. Sun, L. Bradel, C. L. North, and N. Ramakrishnan. The role of interactive biclusters in sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1559–1562. ACM, 2014. doi: 10.1145/2556288.2557337

[18] M. Sun, P. Mi, C. North, and N. Ramakrishnan. Biset: Semantic edge bundling with biclusters for sensemaking. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):310–319, 2016. doi: 10.1109/TVCG.2015.2467813

[19] M. Sun, C. North, and N. Ramakrishnan. A five-level design framework for bicluster visualizations. *IEEE TVCG*, 20(12):1713–1722, 2014. doi: 10.1109/TVCG.2014.2346665

[20] M. Sun, J. Zhao, H. Wu, K. Luther, C. North, and N. Ramakrishnan. The effect of edge bundling and seriation on sensemaking of biclusters in bipartite graphs. *IEEE Transactions on Visualization and Computer Graphics*, 2018. doi: 10.1109/TVCG.2018.2861397

[21] H. Wu, M. Sun, P. Mi, N. Tatti, C. North, and N. Ramakrishnan. Interactive discovery of coordinated relationship chains with maximum entropy models. *ACM Transactions on Knowledge Discovery from Data*, 12(1):7, 2018. doi: 10.1145/3047017

[22] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information Visualization*, 4(4):239–256, 2005. doi: 10.1057/palgrave.ivs.9500099

[23] J. Zhao, M. Sun, F. Chen, and P. Chiu. Bidots: Visual exploration of weighted biclusters. *IEEE transactions on visualization and computer graphics*, 24(1):195–204, 2017. doi: 10.1109/TVCG.2017.2744458