Visualizing Concordance of Sets

Bohyoung Kim[1]
Seoul National University Bundang Hospital,
Gyeonggi, Korea, 463-707

Bongshin Lee[2]
Human-Computer Interaction Lab,
Department of Computer Science,
University of Maryland,
College Park, MD 20742, USA

Jinwook Seo[3]
Children's Research Institute,
Children's National Medical Center,
Washington, DC 20010, USA

Abstract

When there is more than one perspective to interpret a dataset, concordance (or discordance) between the result sets from the different perspectives plays an important role in getting refined results. For example, several clustering algorithms generate different results for the same input. To get useful insights, users need to combine different perspectives by checking concordance between those results. In this paper, we present an interactive visualization tool called ConSet, where users can effectively examine multiple sets at once. It uses permutation matrix visualization to enable users to easily identify similar sets. In addition to a standard Venn diagram, we introduce a Fairy diagram that allows users to compare two or three sets without inconsistencies. We conducted a qualitative user study to evaluate how our tool works in comparison with a traditional set visualization tool based on a Venn diagram. We found that users performed better with ConSet than with the traditional interface for many tasks and most users preferred ConSet.

CR Categories and Subject Descriptors: I.6.9.c Information visualization, H.5.2 User Interfaces, H.5.2.f Graphical user interfaces, H.1.2.a Human factors, H.2.8.c Data and knowledge visualization, H.2.8.h Interactive data exploration and discovery

Additional Keywords: set concordance, Venn diagram, Fairy diagram, Treemap, permutation matrix, cluster comparison, gene ontology

1 Introduction

When there is more than one way to approach a problem, it can be useful to combine multiple perspectives. Visualization of the concordance or discordance of those perspectives can help integrate important knowledge. Scientific problem solving usually involves concordance analysis among several perspectives. This includes problems in information retrieval, bioinformatics, data mining, and so on. For example, Google and MSN search often return different search results. Users could have a more judicious view on the search term by comparing those results. Suppose scientists run an experiment and there are several semi-standard methods to acquire numerical values from a measurement device. The choice of a data acquisition method can profoundly change the resulting data interpretation. Without checking the concordance of different acquisition methods, scientists might have high false positive rates. For example, molecular biologists have to use a "probe set signal algorithm" to acquire signal values of genes from Affymetrix GeneChips. They get different sets of sufficiently powered genes in the subsequent power analysis depending on the signal algorithm used. The concordance of the results sets from different signal algorithms can be checked using set operations in connection with various concordance measures. This enables biologists to identify concordant/discordant genes. Therefore, they can sometimes significantly reduce the false positive rates by simply checking the concordance of the results of different algorithms.

Similar problems occur afterwards. If the biologists decide to use clustering algorithms to identify important patterns in the acquired dataset, they have to decide what kind of clustering algorithms to use. Resorting to only one clustering algorithm could bias the result since different algorithms might come up with completely different patterns depending on how the algorithm detects clusters. Since most clustering algorithms generate disjoint sets(=clusters), there are no similar sets in the result of one clustering algorithm. If we combine all clusters from two clustering algorithms, concordance between those algorithms can be checked by seeing how many sets are similar to each other.

Another example would be when one data element can be classified into multiple categories. For example, a gene product can be related to many gene ontology terms, and a web resource can be mapped to multiple categories in the Open Directory (www.dmoz.org). Identifying the elements classified into different categories helps users unveil the unknown features of the element and of the dataset containing that element.

In existing information visualization tools, brushing and linking techniques [4] were used to show some concordance. Coordinated highlighting of many views for the same dataset can reveal intersection of sets. For example, hierarchical clustering results comparison using paired dendrogram views or phylogenic trees comparison using paired tree views can be thought of as showing concordance of two perspectives, or a group of terminal nodes. Graph visualization can also be a candidate since we can represent each set as a node and the relationship (similarity) of sets as links. While graph drawing techniques combined with clustering approach can show an overview of relationships, such as similarities/dissimilarities among sets, it is not easy to incorporate intuitive ways to support important set operations.

We thought that a more general set visualization tool was necessary to support important tasks for concordance analysis of sets: (1) to show an overview of relationships between sets, (2) to aggregate and filter sets/elements according to users' interests, (3) to efficiently perform fundamental set operations such as intersection and difference, and (4) to generate deeper insight into the original problem from the concordance visualization.

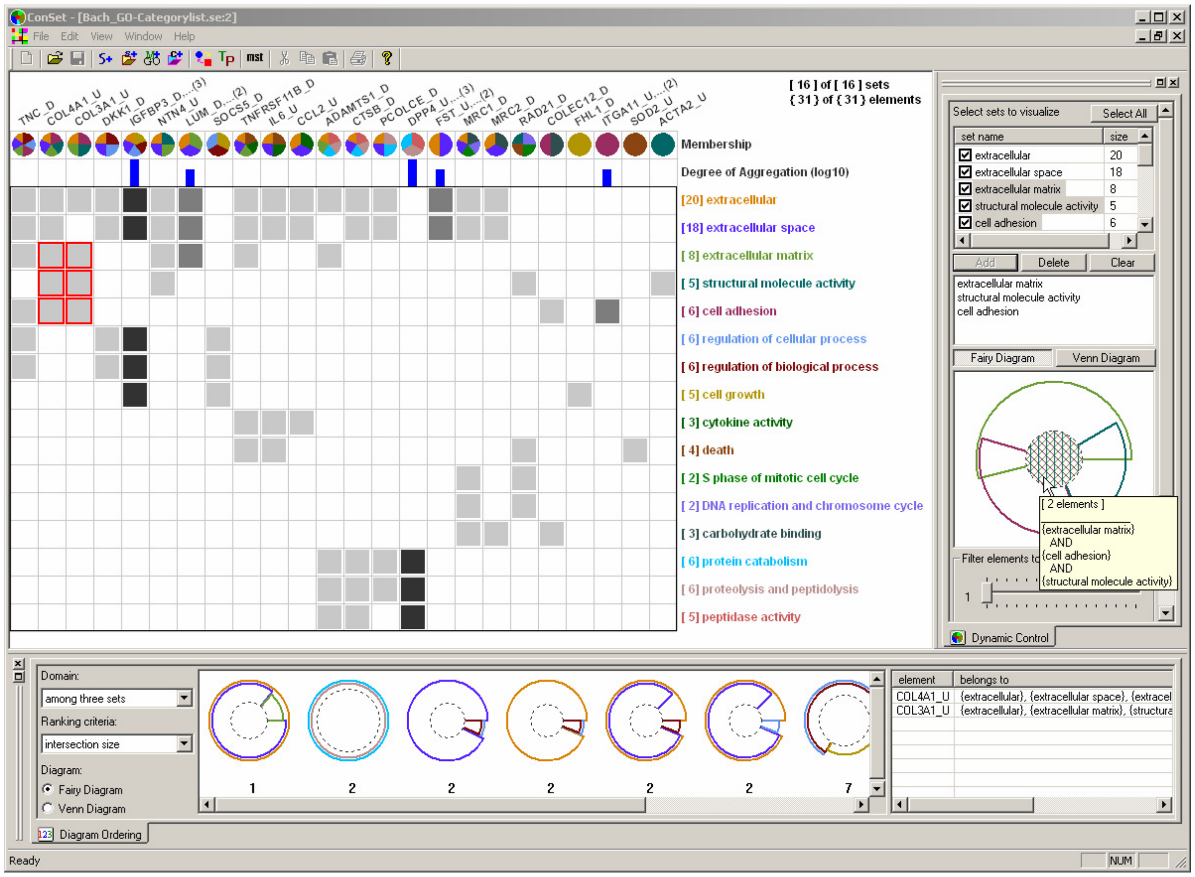
In this paper, we present intuitive interfaces and interactions for set concordance analysis built upon existing visualization techniques such as permutation matrix [6]. The information visualization mantra (overview first, zoom and filter, detail on demand) is the underlying guideline of our ConSet (Figure 1) design.

[1] lemon@infinitt.com
[2] bongshin@cs.umd.edu
[3] jseo@cnmcresearch.org

Report Documentation Page	*Form Approved OMB No. 0704-0188*

1. REPORT DATE **2006**	2. REPORT TYPE	3. DATES COVERED **00-00-2006 to 00-00-2006**
4. TITLE AND SUBTITLE **Visualizing Concordance of Sets**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **human-Computer Interaction Lab,Department of Computer Science,University of Maryland,College Park,MD,20742**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES **The original document contains color images.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**8**	


ConSet - [Bach_GO-Categorylist.se:2]
File Edit View Window Help
[16] of [16] sets
{ 31 } of { 31 } elements
TNC_D COL4A1_U COL3A1_U DKK1_D IGFBP3_D...(3) NTN4_U LUM_D...(2) SOCS5_D TNFRSF11B_D IL6_U CCL2_U ADAMTS1_D CTSB_D PCOLCE_D DPP4_U...(3) FST_U...(2) MRC1_D MRC2_D RAD21_D COLEC12_D FHL1_D ITGA11_U...(2) SOD2_U ACTA2_U
Membership
Degree of Aggregation (log10)
[20] extracellular
[18] extracellular space
[8] extracellular matrix
[5] structural molecule activity
[6] cell adhesion
[6] regulation of cellular process
[6] regulation of biological process
[5] cell growth
[3] cytokine activity
[4] death
[2] S phase of mitotic cell cycle
[2] DNA replication and chromosome cycle
[3] carbohydrate binding
[6] protein catabolism
[6] proteolysis and peptidolysis
[5] peptidase activity
Select sets to visualize
Select All
Add Delete Clear
Fairy Diagram Venn Diagram
Filter elements to
Dynamic Control
Domain:
Ranking criteria:
Diagram:
Diagram Ordering
Ready
NUM

set name	size
☑ extracellular	20
☑ extracellular space	18
☑ extracellular matrix	8
☑ structural molecule activity	5
☑ cell adhesion	6

extracellular matrix
structural molecule activity
cell adhesion

[2 elements]
{extracellular matrix}
AND
{cell adhesion}
AND
{structural molecule activity}

Domain: among three sets
Ranking criteria: intersection size
Diagram: [✓ Fairy Diagram] Venn Diagram

1 2 2 2 2 2 7

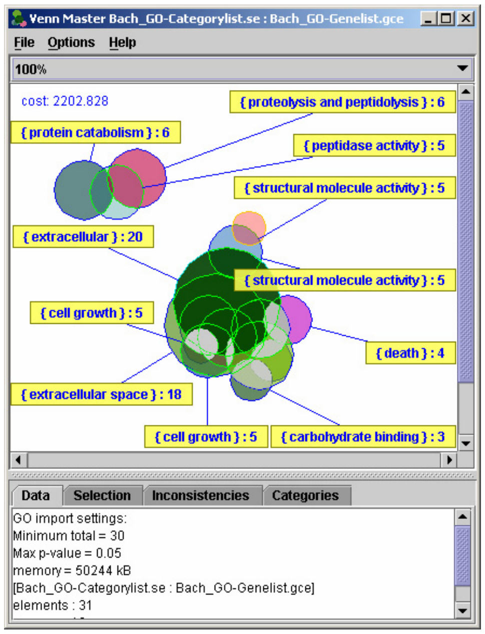
element	belongs to
COL4A1_U	{extracellular}, {extracellular space}, {extracel
COL3A1_U	{extracellular}, {extracellular matrix}, {structura

Figure 1. ConSet with 16 sets and 31 elements. The Permutation Matrix view shows an overview of the relationships among sets and elements. The Dynamic Control view on the right enables users to filter sets and elements. It also allows users to select two or three sets to show a diagram. The Diagram Ordering view at the bottom shows the top 10 diagrams of two or three sets.

Users can see an overview of relationships among sets and elements in a permutation matrix. Aggregation of elements with the same membership and filtering by dynamic query devices enable users to narrow down to a handful of important sets and elements. Users can also see the relationships between two or among three sets in the conventional Venn diagram or our novel Fairy diagram. We also ran a qualitative user study in comparison with VennMaster (shown in Figure 2), a set visualization tool that uses a generalized Venn diagram. The user study suggested that ConSet supports more tasks with less errors compared to VennMaster. The user study also enabled us to identify several ways to improve the interface and design of ConSet.

2 Related Work

Brushing and linking, powerful information visualization techniques, can be used to reveal concordances between sets. Coordinated multiple views provide users with ways to understand relationships between datasets behind the views [3]. HCE shows two dendrograms at once, highlights the corresponding terminal nodes in the two dendrograms, and shows the mapping with connecting lines when users click on a branch of a dendrogram [13]. TreeJuxtaposer [12] also applies the brushing and linking techniques as well as Focus+Context techniques [8] to compare two large phylogenic trees with guaranteed visibility. Users can see the discordance of the two hierarchical structures by examining the highlighting and/or connections. Sometimes, the main purpose of selecting an internal node on a tree visualization is to select a set of terminal nodes reachable from the internal node. This problem can be generalized as a set visualization and the main task can be concordance checking among sets.


Venn Master Bach_GO-Categorylist.se : Bach_GO-Genelist.gce
File Options Help
100%
cost: 2202.828
{ proteolysis and peptidolysis } : 6
{ protein catabolism } : 6
{ peptidase activity } : 5
{ structural molecule activity } : 5
{ extracellular } : 20
{ structural molecule activity } : 5
{ cell growth } : 5
{ death } : 4
{ extracellular space } : 18
{ cell growth } : 5
{ carbohydrate binding } : 3
Data Selection Inconsistencies Categories

```
GO import settings:
Minimum total = 30
Max p-value = 0.05
memory = 50244 kB
[Bach_GO-Categorylist.se : Bach_GO-Genelist.gce]
elements : 31
```

Figure 2. VennMaster with the same dataset as in Figure 1. We manually placed labels of some sets using VennMaster.

MetaCrystal [17] is a visualization tool based on the InfoCrystal layout [16] that helps users fuse together search results from different search engines. It utilizes various visual features such as shape, size, color, proximity, and orientation to show the degree of overlap among different search results. Overlapping search results are expected to provide more comprehensive, relevant, and effective view on the subjects delivered by the search terms. Here again, users' tasks performed with MetaCrystal can also be thought of as concordance checking among sets (i.e. search results from different search engines).

When it comes to set visualization, Venn diagrams are the de facto standard. A Venn diagram is a special case of an Euler diagram. Venn diagrams should have areas to represent all possible combinations of sets regardless of whether that area is actually empty or not. This restriction is loosened in Euler diagrams, where empty areas do not have to appear. These diagrams are applied to various problems in bioinformatics, information retrieval, and information visualization. New applications sometimes require some additional restrictions on how to draw Euler diagrams such as the one that the shape of contour should be a circle and more information such as cardinality is coded as size (area) and/or color of a contour. It is important to mention that the terms Venn diagram and Euler diagram are often confused. Euler diagrams, where each contour is a circle, are often called Venn diagrams, even though theoretically this is not correct. In this paper, we follow this general perception of Euler diagram and use the term Venn diagram for the Euler diagram with the constraint.

VennMaster is to our knowledge the only visualization tool that shows an arbitrary number of sets in Venn diagrams, where each set is represented as a polygon with a user-defined number of edges [10]. When there are enough edges, each set appears almost like a circle. The size of each polygon is proportional to the cardinality of the corresponding set. All properly size-coded polygons are placed in such a way that the size of each intersection area is also proportional to the number of elements in the intersection. Since the optimal size coding and layout determination are too expensive to be solved in a pure analytical way, they resort to genetic algorithm techniques.

VennMaster was developed to improve users' interpretation and visualization of the output of a famous bioinformatics tool, or GoMiner. GoMiner enables researchers to query the gene ontology database (www.geneontology.org; comprehensive annotation of genes or gene products) for associated categories in a cellular context [19]. Since one gene can be associated with more than one gene ontology category, the interpretation of such complex associations is a challenging task. VennMaster translated this problem into a set relationship visualization problem (i.e., treating a gene ontology category as a set and a gene product as an element). Since the approach was very useful, VennMaster was integrated into GoMiner.

While it is useful to have one more visualization approach adopted to a well-known bioinformatics tool, this approach still has a lot of drawbacks from an information visualization perspective. First of all, there are three kinds of inconsistencies in the VennMaster visualization: (1) it is not guaranteed that all possible intersections are visible in the generalized Venn diagram display, so those so-called inconsistent intersections are shown in a separate list view, (2) since it uses regular convex polygons, there will be intersections of polygons where no element is mapped, which will be explained later in the next section, and (3) the resulting layout of diagrams can be different in each run of the program because it uses a genetic algorithm to optimize the layout.

A matrix-based representation was often used to show relationships between items by using both rows and columns to represent items and values in each cell to show the relationship. For example, Abello and Korn presented matrix and color map based techniques to visualize phone calls made between states [1]. Van Ham used multilevel call matrices in the management of large software projects [18]. Kincaid applied an extended permutation matrix to the task of exploratory data analysis of multi-experiment microarray studies [11]. Ghoniem *et al.* used adjacency matrices to interactively visualize and explore relations between constraints and variables in constraint problems [9].

We thought that information visualization techniques could improve users' experience in interpreting such complex set relationships. It can be accomplished without the overburden of drawing a lot of circles in proper scale and location. Furthermore, we can maintain the familiarity of simple diagrams such as Venn diagrams. We applied the permutation matrix display to set concordance visualization to provide a better overview of set concordance without inconsistencies mentioned above. Interactive selection and filtering methods enable users to narrow down to a handful number of sets. The detail is shown as a general Venn diagram or our new Fairy diagram after users select two or three sets.

3 VISUALIZING CONCORDANCE OF SETS

3.1 Untangling Overlaps

While significant overlaps of many sets in the general Venn diagram visualization tool clearly shows high similarities of sets, those overlaps make it difficult to see the details on memberships of elements to sets. In addition, non-overlapped areas are hard to select when overlaps cover most of the elements. We thought that a permutation matrix, a proven multidimensional visual structure, could help untangle overlaps while carrying similarity information. For the set concordance visualization, each column represents an element and each row represents a set (Figure 3). If an element e_j belongs to a set S_i, we fill the cell $C(i, j)$ with gray, otherwise $C(i, j)$ is empty. Each set is given a distinctive color and the set name is displayed at the end of its corresponding row in its own color.

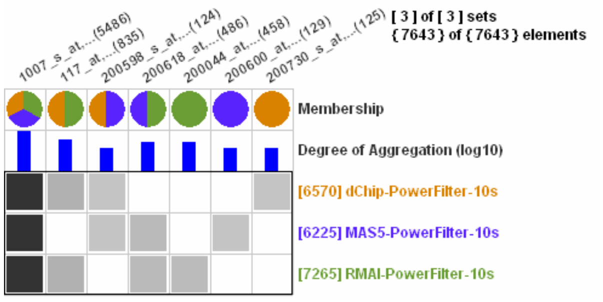


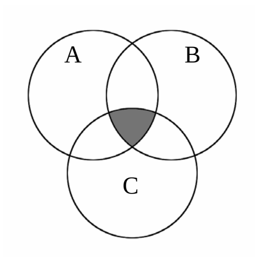
Figure 3. Permutation matrix view for set concordance visualization shows the concordance of three power analysis results by three probe set signal algorithms with 7643 genes. Aggregation drastically reduced the number of columns from 7643 to 7. The degree of aggregation is shown as histograms in log scale.

We show information regarding elements at top three rows, which we call "column header." The column header includes, from top to bottom, *Element Name*, *Membership*, and *Degree of Aggregation*, each in a separate row. The membership row shows pie chart-like glyphs, where each pie represents a set to which the corresponding element belongs and is filled with the color of the set. With the color-coded membership information, users can easily grasp how many sets an element belongs to.

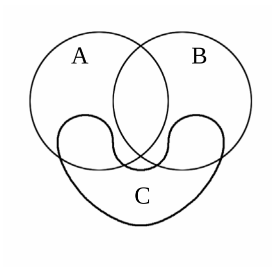
Since all elements are visible unlike Venn diagram visualizations, it is necessary to implement a method to accommodate a large number of columns. It is reasonable to assume that many elements would share the same membership, when the number of elements is significantly larger than the number of sets. Thus, by aggregating those elements into a single column, it is possible not only to save a significant amount of screen space but also to have a clear overview in a compact form. When several elements are aggregated to a single column, only the representative element is shown in the permutation matrix, and other aggregated elements are hidden. The name of the column is the representative element that comes first in alphabetical order. The number of aggregated elements is given in parentheses at the end of the representative element's name. In addition, the number of aggregated elements is visualized as a blue bar in the *Degree of Aggregation* row. The height of each bar is proportional to the number of the aggregated elements, and users can show the bars in log scale. The intensity of a cell in the permutation matrix is also proportional to the number of aggregated elements.

3.2 Avoiding Inconsistencies

Venn diagrams are widely used to represent set relationships. While they are intuitive and familiar to users, Venn diagrams have the drawback of inconsistencies: missing valid intersection areas and showing invalid intersection areas. First, let's assume relationships among three sets *A*, *B* and *C*, where *A* and *B* have some common elements and *C* has elements in $(A-B)$ and $(B-A)$ but not in $(A \cap B)$. If we represent this relationship in a Venn diagram, an empty set $(A \cap B \cap C = \phi)$ is shown as a region in gray (Figure 4a). If we loosen the constraint that each set should be a circle, this relationship can be represented in a Venn diagram without such inconsistency (Figure 4b). Then, however, the diagram loses the advantage that users are used to it. The other inconsistency is incurred by the fact that it is very hard to achieve a valid Venn diagram when the number of sets is large. Furthermore, it is almost impossible to accurately size-code all possible zones. Thus, it is common that some valid intersection areas are missing in Venn diagrams especially when many sets have intersections with many others.



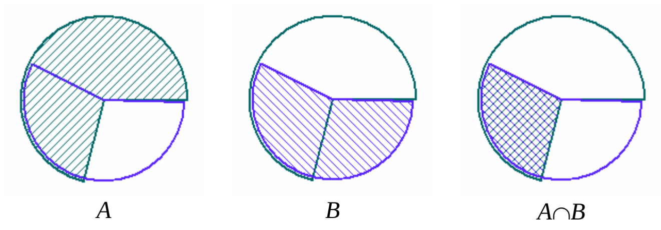
(a) Venn Diagram with inconsistency



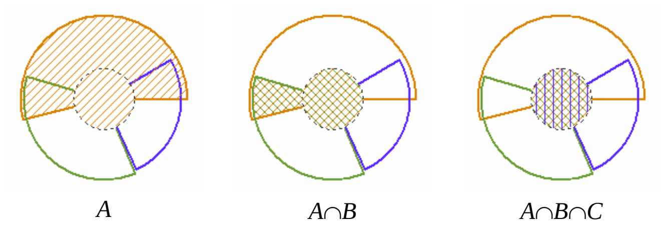
(b) Euler Diagram without inconsistency

Figure 4. Inconsistency of Venn Diagrams. (a) and (b) show the same sets relationships. There is no element in the gray area at (a), but there is no way to avoid this inconsistency in Venn Diagrams. By allowing C to have a concave contour, it is possible to avoid the inconsistency in Euler Diagrams (b).

To maintain users' familiarity with Venn diagrams while avoiding the two inconsistencies, we suggest applying the information visualization mantra (overview first, zoom-and-filter, detail on demand) [15]. We use a permutation matrix view to show an overview. Dynamic queries, manual selections, and ranking of sets allow users to narrow down to two or three sets to have an easy-to-understand diagram. However, even with three sets, Venn diagrams still suffer from the two inconsistencies explained above. Thus, we propose a new diagram named Fairy diagram shown in Figure 5. A Fairy diagram does not contain any invalid intersection areas and all areas are accurately size-coded by the number of elements in regions. It looks like a roulette wheel, where each set is represented as a fan.



(a) Two Sets



(b) Three Sets

Figure 5. Fairy Diagram

For two sets *A* and *B*, a circle represents the union $(A \cup B)$. The center angle of the fan for *A* is calculated as follows.

$$\theta_A = 2\pi \times \frac{n(A)}{n(A \cup B)},$$

where $n(A)$ is the cardinality of set *A*. The center angle of the fan for *B* is calculated in the same way. If the intersection $(A \cap B)$ is not empty, the two fans for *A* and *B* overlap. The center angle for the overlapping fan is calculated as follows.

$$\theta_{(A \cap B)} = 2\pi \times \frac{n(A \cap B)}{n(A \cup B)}$$

Therefore, all regions split by the fans of the sets *A* and *B* are accurately size-coded.

For three sets *A*, *B* and *C*, a circle represents the union $(A \cup B \cup C)$. The intersection $(A \cap B \cap C)$ is represented as a center circle. If the outer circle has the radius of *R*, the radius of the center circle (*r*) is calculated as follows.

$$\frac{n(A \cap B \cap C)}{n(A \cup B \cup C)} = \frac{\pi r^2}{\pi R^2} \qquad \therefore r = \sqrt{\frac{n(A \cap B \cap C)}{n(A \cup B \cup C)}} \times R$$

Thus, the area of the center circle for the set $(A \cap B \cap C)$ is exactly proportional to the cardinality of $(A \cap B \cap C)$. A doughnut-shaped region between the center and outer circles represents the set $((A \cup B \cup C) - (A \cap B \cap C))$. In the doughnut-shaped region, there are three doughnut segments for the three sets $A-(A \cap B \cap C)$, $B-(A \cap B \cap C)$, and $C-(A \cap B \cap C)$. Each doughnut segment has a center angle in proportion to the cardinality of the corresponding set. The center angle of the doughnut segment for the set $(A-(A \cap B \cap C))$ is calculated as follows.

$$\theta_{(A-(A \cap B \cap C))} = 2\pi \times \frac{n(A) - n(A \cap B \cap C)}{n(A \cup B \cup C) - n(A \cap B \cap C)}$$

Thus, we can accurately size-code all regions split by the center and outer circles and three doughnut segments.

While Fairy diagrams have advantages such as no inconsistencies and accurate size-coding as shown above, there are some problems with this approach. For example, circles and doughnut-shape regions are, in theory, drawn within a circle and a part of some outer arcs can overlap each other. Thus, sometimes it is difficult to know the exact bounds of a region. This problem can be attenuated by drawing region boundaries with a tiny displacement as shown in Figure 5.

3.3 Ordering Sets and Elements

The ordering of columns and rows significantly influences the pattern of a permutation matrix. Generally, the goal of reordering in a permutation matrix is to move significant cells to the diagonal of the matrix [7]. Since this is not eligible in our permutation matrix for set concordance visualization, we propose three different ordering methods. First, we suggest Minimum-Cost Spanning Tree (MST) ordering, where elements of similar membership are placed close to each other. To apply Prim's algorithm [2], one of the MST construction algorithms, each element is represented as a vertex and the relationship between every pair of elements is represented as an edge whose cost is inversely proportional to how similar their memberships are. The cost between two elements is calculated as follows.

$$Cost(e_m, e_n) = 1 - \frac{(\#\text{ of sets with both } e_m \text{ and } e_n)}{(\#\text{ of all sets})}$$

As the number of sets that have both e_m and e_n increases, the cost becomes smaller (i.e., two elements are more similar). After calculating costs of every pair of elements, Prim's algorithm is used to build a minimum cost spanning tree. The vertex corresponding to an element that belongs to the most sets is served as the start vertex. Next vertex to be added is the vertex that is not in the current spanning tree and is closest to some vertex in the tree. According to the sequence that vertices are added in the algorithm, the corresponding elements are ordered.

Sets are ordered in the same way as elements except for the cost function, which is defined as follows.

$$Cost(S_i, S_j) = 1 - \frac{n(S_i \cap S_j)}{n(S_i \cup S_j)}$$

Since MST ordering of sets and elements significantly improve the permutation matrix, concordance among sets and even among elements can be examined more efficiently.

While MST ordering helps users identify similar elements and sets, more ordering methods are useful to support other tasks such as finding a specific element or the biggest set. For example, it is easier to find an element when elements are in alphabetical order. We provide three additional ordering methods for elements: move a column to the right end, order by name, and order by the number of memberships; and two more for sets: order by name and cardinality.

4 CONSET INTERFACE

We developed a visualization tool named ConSet by applying design ideas described in the previous section. ConSet enables users to examine the concordance of sets visually and interactively. ConSet consists of three views; Permutation Matrix, Dynamic Control, and Diagram Ordering views (Figure 1). The Permutation Matrix view shows an overview of all the visible sets. The Dynamic Control view on the right contains the sets list, the diagram area and the filter controls. The Diagram Ordering view at the bottom has the ranked diagrams area and the elements list.

4.1.1 Easy Access of Sets and Elements

ConSet, by default, rearranges the sets by the MST ordering. Since this places sets with more common elements closer to each other, users can easily find similar sets. In addition, the sets can also be ordered by their name and cardinality, which is available on the sets list in the Dynamic Control view.

ConSet also provides four element re-ordering methods. When users right-click the mouse on a column header, a pop-up menu for element re-ordering shows up. Selecting a menu item, users can move elements to the right end of the column. This enables users to easily compare several elements of interest by putting them side by side and right next to the set names. Similar to sets, elements can also be sorted by three criteria; alphabetically, by the number of memberships, and by MST ordering.

When users move the mouse over a column header of an element, ConSet highlights the corresponding column with a greenish-gray rectangle. In addition, the names of sets that do not contain that element are grayed out. This helps users identify all the sets that an element belongs to. The name of the element is also shown in the elements list in the Diagram Ordering view along with their membership information. If the column is aggregated, the names of all the aggregated elements are shown.

Similarly, if users move the mouse over a set name, the corresponding row is highlighted with a rectangle in the set's own color. The names of elements that do not belong to the selected set are grayed out. The names of all the elements of the highlighted set come in the elements list. If users move the mouse over a gray-filled cell $C(i, j)$ in the Permutation Matrix view, the cell is highlighted by a red rectangle with the j-th element's name highlighted in red and the i-th set's name underlined in red. The name of the j-th element and the names of its aggregated, if any, elements are shown in the elements list.

4.1.2 Dynamic Filtering of Sets and Elements

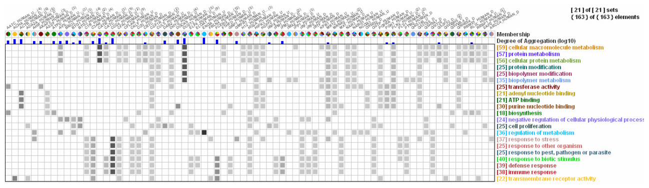
ConSet, by default, shows the names of all the sets in the sets list in the Dynamic Control view. It allows users to change the visibility of sets in the Permutation Matrix view. For example, if users check (or uncheck) a check box right before a set name in the sets list, ConSet shows (or hides) the set in the Permutation Matrix view. This enables users to identify similar ones among the sets of their interest. For example, the number of sets was reduced from 21 (Figure 6a) to 10 (Figure 6b) when we hid the sets whose cardinality is less than 30. The aggregation of elements is based on their memberships to the visible sets, not to all the sets. So, whenever the visibility of sets changes, ConSet re-computes aggregation.

ConSet also enables users to filter elements to be shown in the Permutation Matrix view. For example, the "Filter elements to show" slider control with a value t filters to show only elements that belong to at least t sets. Filtered elements or sets can either be removed from or be grayed out in the Permutation Matrix view. The number of elements was further reduced from 133 (Figure 6b) to 24 (Figure 6c) when we filtered out the elements that do not belong to at least 5 sets.

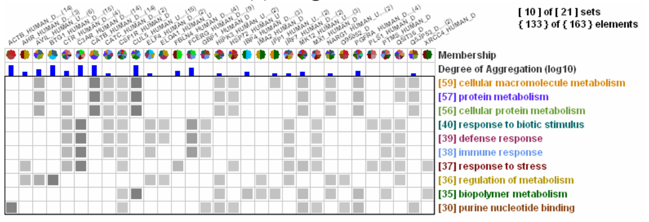
4.1.3 Showing Relationships between Sets

ConSet visualizes the relationship of two or three sets in the diagram area in the Dynamic Control view. Users can add up to three sets into the diagram area from the sets list. When users select a set in the sets list, the corresponding set is highlighted in the Permutation Matrix view while the names of all the elements of the selected set are shown in the elements list in the Diagram Ordering view. When they click the "Add" button at the bottom of the sets list, selected sets are added to the diagram area. The names of added sets are displayed in the upper window of the diagram area and a diagram of their relationship is drawn in the lower window of the diagram area. Users can remove sets from the diagram area by clicking the "Delete" button after selecting them from the upper window. They can also clear the diagram area by clicking the "Clear" button.

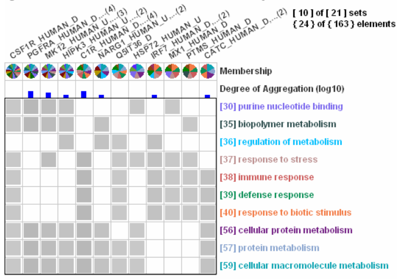
When users move the mouse over a set in a Venn diagram or a Fairy diagram, a tooltip appears to show its name and cardinality. At the same time, the set is highlighted in the Permutation Matrix view and the elements information in the set is shown in the elements list. When users move the mouse over a region for an intersection, the elements in the intersection are highlighted in the Permutation Matrix view and their information appears in the elements list. If users click on a region in a diagram, the corresponding region is selected and the selection is toggled on another click. This enables users to examine all elements in the elements list when scrolling is required.



(a) Original data


[10] of [21] sets
{ 133 } of { 163 } elements
Membership
Degree of Aggregation (log10)
[59] cellular macromolecule metabolism
[57] protein metabolism
[56] cellular protein metabolism
[40] response to biotic stimulus
[39] defense response
[38] immune response
[37] response to stress
[36] regulation of metabolism
[35] biopolymer metabolism
[30] purine nucleotide binding

(b) Filtering out sets whose cardinality is less than 30 from (a)


[10] of [21] sets
{ 24 } of { 163 } elements
Membership
Degree of Aggregation (log10)
[30] purine nucleotide binding
[35] biopolymer metabolism
[36] regulation of metabolism
[37] response to stress
[38] immune response
[39] defense response
[40] response to biotic stimulus
[56] cellular protein metabolism
[57] protein metabolism
[59] cellular macromolecule metabolism

(c) Filtering out elements that belong to less than 5 sets from (b)

Figure 6. Sets and Elements Filtering with Human Muscular Dystrophy Dataset of 21 sets and 163 elements.

4.1.4 Diagram Ordering using the Rank-by-Feature Framework

We applied the Rank-by-Feature Framework [14] to ConSet. The Diagram Ordering view shows the top 10 diagrams ranked by some criteria. From the "Domain" combo-box at the top left corner of the view, users can select the ordering of diagrams between two or among three sets. Two ranking criteria are provided in the "Ranking criteria" combo-box. The criterion "intersection size" ranks diagrams by the size of the intersection, and the criterion "overlap metric" orders diagrams by the ratio of the intersection set size to the union set size. This helps users easily capture a collection of important sets that meets the ranking criteria. Users can see each of the top 10 ranked diagrams in two ways; Venn diagram and Fairy diagram. They work the same as in the diagram area of the Dynamic Control view.

5 OTHER APPLICATION EXAMPLES

We extended ConSet to help users compare clustering results by adding special functionality. An output of a clustering algorithm is in most cases a group of disjoint clusters(=sets), each of which is a set of elements. ConSet arranges sets forming several groups where a set from one clustering result is put together with one or more similar sets from the other clustering result. ConSet arranges these groups row by row and adds a special row (*Cluster Concordance*) right before the first group, where all matching elements within a group are projected and color coded by the ratio of the cardinalities of two matching sets. This color coding is intended to give proper penalty to the cases where one big cluster from one clustering result overlaps with several small clusters from the other clustering result, which is not so interesting concordance.

Figure 7 and Figure 8 visualize the concordance between the hierarchical clustering result and K-means clustering result with Euclidean distance measure with 77 breakfast cereals data and with Census data of 224 US eastern counties near MD, respectively. Many dense red cells at the *Cluster Concordance* row in Figure 7 indicate that those two results are very concordant with each other despite an outlier, "Multigrain_Cheerios," which does not belong to any matching clusters pair. On the other hand, Figure 8 shows that, overall, the two clustering results for the census data set are not so concordant even though there are several strong matching counties groups with dense red cells on the *Cluster Concordance* row.

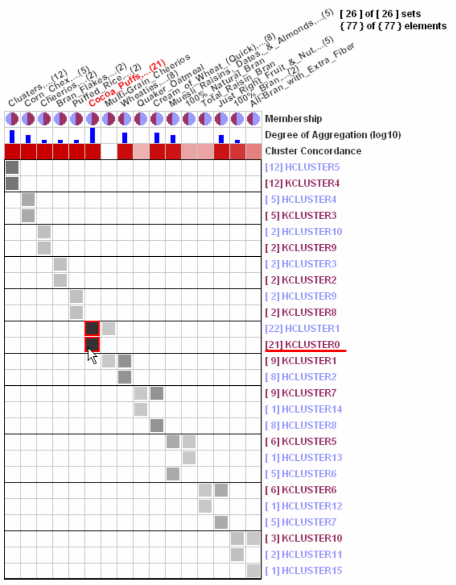


Figure 7. Clustering Results Comparison (HCLUSTER: Hierarchical Clustering, KCLUSTER: K-means Clustering) with 77 breakfast cereals data

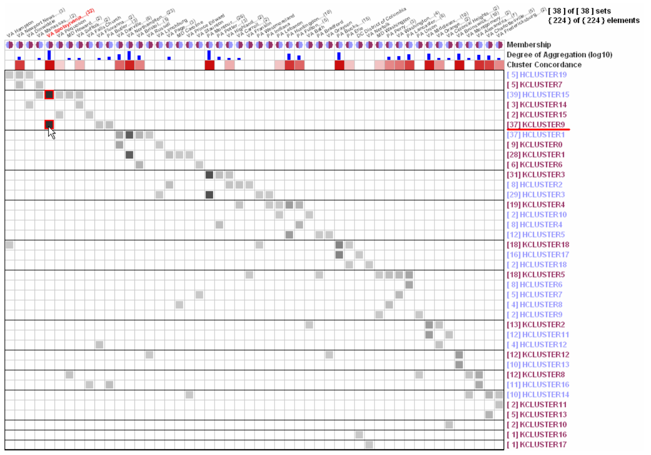


Figure 8. Clustering Results Comparison (HCLUSTER: Hierarchical Clustering, KCLUSTER: K-means Clustering) with Census data of 224 US eastern counties near MD

The same approach can also help users identify statistical associations between categorical variables or between a clustering result and a categorical variable. Users can partition a dataset into disjoint sets according to a categorical variable. For example, the census data for all US counties can be partitioned into disjoint sets according to categorical variables, such as "poverty level" and "education level." Since an integer- or real-type variable can be converted to a categorical variable by a simple binning, ConSet can visualize statistical associations between a categorical variable and an integer- or real-type variable.

6 ConSet Evaluation

We conducted a qualitative study to understand how well ConSet works and to identify any usability issues. We originally wanted to compare three approaches – Treemap layout [5] (shown in Figure 9), permutation matrix, and VennMaster. However, from our own experience with ConSet, we suspected that Treemap layout approach would work best only for identifying the biggest sets. Since this task can be easily completed by other approaches with a sorting feature, we decided to compare ConSet only with permutation matrix to VennMaster. We measured the time to complete each task using a stopwatch and counted the number of wrong answers, time-outs, and give-ups. The experimenter also took notes on usability issues participants experienced during the walk through of the system.

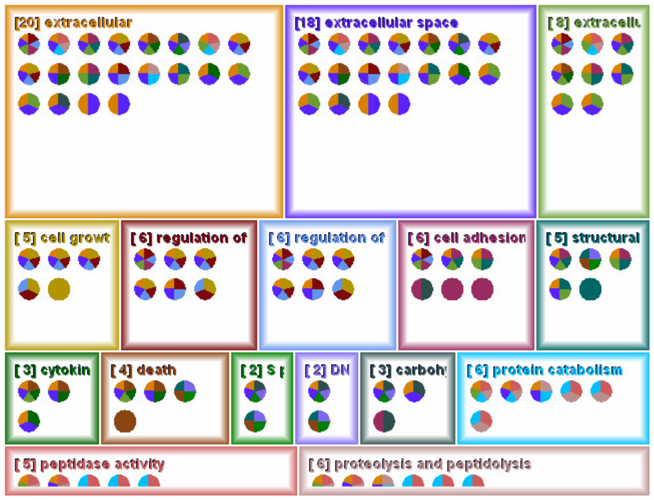


Figure 9. ConSet with Strip Treemap Layout. Each box represents a set, which has a unique border color. Set name and its cardinality are shown at the top left corner of each box. The box size is proportional to the number of elements in the set.

6.1 Data and Participants

We used two similar datasets exported from GoMiner for this user study. One dataset had 16 sets 31 elements and the other had 23 sets and 28 elements. ConSet is implemented to import the pair of GoMiner's category summary file and gene summary by category file. From the pair of files, ConSet builds a number of sets of genes, each of which is a gene ontology category.

We recruited 8 biologists (5 males and 3 females) including 1 mail pilot subject. The pilot data is not included in the reporting of the experimental task data because the interfaces and tasks were improved after the pilot.

6.2 Procedure and Tasks

Each participant used both interfaces; interface order was counterbalanced. Participants first received training on the first interface and were allowed to play with the program to learn the basic features. They were allowed to ask questions during the training. For each interface, participants spent about 10 minutes on average. Next, they were asked to conduct 9 tasks as quickly as they were possible. Each task had a 3-minute time limit and participants were allowed to give up a task at any time. After a short break, the same procedure was repeated with the second interface. Preferences, comments, and suggestions were collected during debriefing. Each session lasted 38 minutes on average. The list of tasks follows.

1. What are the top three biggest sets?
2. What is the size of the biggest set?
3. What are the top three elements that belong to the most sets?
4. Name the sets that have a given element.
5. Name the sets that have two given elements.
6. What are three sets that share the most elements?
7. Name the elements in the intersection of two sets?
8. Name the elements in the intersection of three sets?
9. Name the elements that are in A but not in B.

6.3 Results

6.3.1 Task times, Error, and Preferences

Out of 63 questions across participants, while there were only 6 time outs and 5 incorrect answers with ConSet, there were 30 time outs and 10 incorrect answers with VennMaster. For task 6, two participants forgot how to use diagram ordering in ConSet. Two participants were not able to complete for task 9 and one for task 1 and 5. Task completion times for time outs were not included in the task time analysis.

As can be seen from Figure 10, participants completed most tasks faster with ConSet. In fact, with VennMaster no one could complete task 3, 4, and 5 within the 3 minute time limit. However, 7, 6, and 5 participants answered correctly with ConSet for task 3, 4, and 5 respectively. We believe this is because ConSet provides good support for showing the names of elements.

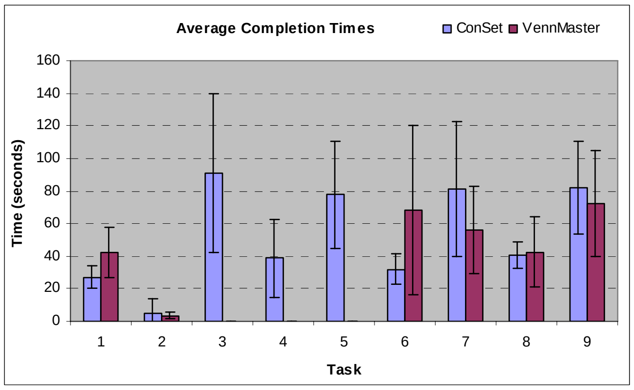


Figure 10. Average task completion times

When asked which interface they preferred overall, 6 out of 7 participants chose ConSet over VennMaster. The reasons from participants include "I was able to complete all tasks," "I like interactive highlighting," "more user-friendly," and so on. One participant who preferred VennMaster said that it is simple and she got used to it. She also said that she might change her preference if she gets comfortable with the Permutation Matrix view by using it more. And one other participant who preferred ConSet said that more training time is needed to get used to ConSet.

6.3.2 Usability Issues

We observed several usability issues in ConSet that needed to be addressed. There was clear user frustration around the selection of sets in the Dynamic Control view on the right. Three participants had a difficulty choosing sets to show in the diagram view. Even though the check box in front of the set name is to filter sets to show in the main Permutation Matrix view, some of the participants thought that the checked sets would be added into the diagram area.

Another issue is that there is no way to select the difference area (A – B). This is because single click behaves differently depending on where users select; click on the intersection area selects the intersection but click on the difference selects the entire set. To address this issue, we can introduce more consistent interaction style to select areas in the Venn and Fairy diagrams. First, single click selects the smallest containing area. So, if users click on the difference or intersection area, the difference or intersection will be selected. Second, users can combine two areas by clicking an area with the control key. Lastly, double click on an area selects all the sets that contain the area. So, users can select an entire set by double clicking on the difference area.

There is no efficient way to find elements/sets with their name. Even though ConSet enables users to sort elements/sets by their name, four participants did not use the sort feature and sequentially scan element names for task 4. This would be a bigger problem when the number of elements is large. We can address this issue by providing a simple search on the element and set name.

The familiarity with the traditional Venn diagram makes it hard for users to utilize a new Fairy diagram. In addition, the task used in the study was easy enough to be completed with the Venn diagrams. However, we believe that instantaneous highlighting of the area on mouse-over along with informative tooltip text helped users understand how to interpret the diagram. It was encouraging to observe some users utilized the Fairy diagram after a short tutorial.

7 Conclusion

We developed a general set visualization tool called ConSet built upon the permutation matrix, which supports important tasks for concordance analysis of sets and elements. ConSet shows an overview of relationships among sets and helps users efficiently perform fundamental set operations such as intersection and difference. ConSet provides the top 10 collections of sets that are most similar, measured either by the number of common items or by the overlap metric. ConSet also enables users to aggregate and filter sets/elements, which improves the scalability.

Our Fairy diagram addresses the two inconsistency problems that may incur in Venn diagrams: missing valid intersection areas and showing invalid intersection areas. It also provides exact size coding of all areas. And intersection of three sets is clearly visualized as a center circle. Permutation matrix display makes it possible to avoid the problem that too many sets overlap in the general Venn diagrams. Another strength of the permutation matrix is that it provides better support for showing the names of elements. ConSet performed much better when tasks required users to access information through elements.

We conducted a qualitative user study to evaluate how our tool works in comparison with a traditional set visualization tool based on a Venn diagram. We found that users performed better with ConSet than with the traditional interface for many tasks and most users preferred ConSet.

Acknowledgement: We would like to thank Catherine Plaisant and Aaron Clamage for their thoughtful comments. This work was supported by Department of Defense W81XWH-04-01-0081 and NIH 1P30HD40677-01 (MRDDRC Genetics Core).

Referernce

[1] J. Abello and J. Korn, "MGV: A System for Visualizing Massive Multigraphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, pp. 21-38, 2002.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*. Reading, MA: Addison-Wesley, 1983.

[3] M. Baldonado, A. Woodruff, and A. Kuchinsky, "Guidelines for Using Multiple Views in Information Visualization," in *Proceedings of Advanced Visual Interfaces*, pp. 110-119, 2000.

[4] R. A. Becker and W. S. Cleveland, "Brushing Scatterplots," *Technometrics*, vol. 29, pp. 127-142, 1987.

[5] B. B. Bederson, B. Shneiderman, and M. Wattenberg, "Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies," *ACM Transactions on Graphics*, vol. 21, pp. 833-854, 2002.

[6] J. Bertin, *Graphics and Graphic Information-Processing*. Berlin; New York: de Gruyter, 1981.

[7] S. K. Card, J. D. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*. San Francisco, CA: Morgan-Kaufmann, 1999.

[8] G. W. Furnas, "Generalized Fisheye Views," in *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 18-23, 1986.

[9] M. Ghoniem, N. Jussien, and J.-D. Fekete, "VISEXP: Visualizing Constraint Solver Dynamics Using Explanations," in *Proceedings of the Seventh International Florida Artificial Intelligence Research Society Conference*, 2004.

[10] H. A. Kestler, A. Muller, T. M. Gress, and M. Buchholz, "Generalized Venn Diagrams: A New Method of Visualizing Complex Genetic Set Relations," *Bioinformatics*, vol. 21, pp. 1592-1595, 2005.

[11] R. Kincaid, "VistaClara: An Interactive Visualization for Exploratory Analysis of DNA Microarrays," in *Proceedings of the Symposium on Applied Computing*, pp. 167-174, 2004.

[12] T. Munzner, F. Guimbretiere, S. Tasiran, L. Zhang, and Y. Zhou, "TreeJuxtaposer: Scalable Tree Comparison using Focus+Context with Guaranteed Visibility," *ACM Transactions on Graphics*, vol. 22, pp. 453-462, 2003.

[13] J. Seo and B. Shneiderman, "Interactively Exploring Hierarchical Clustering Results," *Computer*, vol. 35, pp. 80-86, 2002.

[14] J. Seo and B. Shneiderman, "A Rank-by-Feature Framework for Interactive Exploration of Multidimensional Data," *Information Visualization*, vol. 4, pp. 99-113, 2005.

[15] B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," in *Proceedings of IEEE Symposium on Visual Languages*, pp. 336-343, 1996.

[16] A. Spoerri, "InfoCrystal: a Visual Tool for Information Retrieval," in *Proceedings of the IEEE Visualization Conference*, pp. 150-157, 1993.

[17] A. Spoerri, "MetaCrystal: Visualizing the Degree of Overlap Between Search Engines," in *Proceedings of the ACM International World Wide Web Conference*, pp. 378-379, 2004.

[18] F. van Ham, "Using Multilevel Call Matrices in Large Software Projects," in *Proceedings of IEEE Symposium on Information Visualization*, pp. 227-232, 2003.

[19] B. Zeeberg, W. Feng, G. Wang, M. Wang, A. Fojo, M. Sunshine, S. Narasimhan, D. Kane, W. Reinhold, S. Lababidi, K. Bussey, J. Riss, J. Barrett, and J. Weinstein, "GoMiner: A Resource for Biological Interpretation of Genomic and Proteomic Data," *Genome Biology*, vol. 4, pp. R28, 2003.