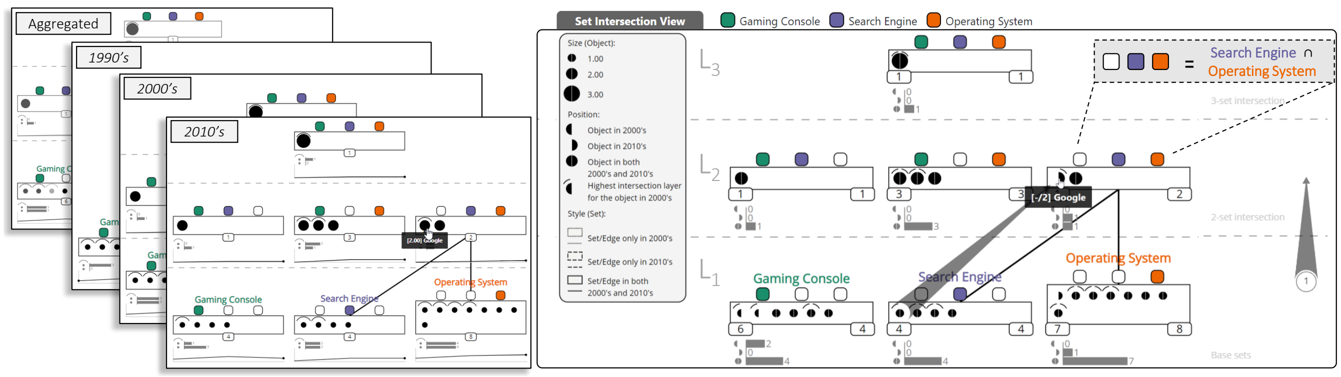
Vision, Modeling, and Visualization (2020)
J. Krüger, M. Niessner, and J. Stückler (Eds.)

Visualizing Sets and Changes in Membership Using Layered Set Intersection Graphs

Shivam Agarwal[1]†, Gleb Tkachev[2]‡, Michel Wermelinger[3]§, and Fabian Beck[1]¶

[1] University of Duisburg-Essen, Germany, [2] University of Stuttgart, Germany, [3] Open University, UK



(a) Aggregated and static set intersection graphs (b) *Diff* view between timesteps *2000's* and *2010's*

Figure 1: *Layered set intersection graphs on sample data about types of products (sets) made by companies (set elements). Each color represents a set, while their combinations identify set intersections. Filled black circles show membership of elements. The visualization can show (a) aggregated or individual static set intersection graphs and (b) differences in set intersection graphs between any two timesteps.*

Abstract

Challenges in set visualization include representing overlaps among sets, changes in their membership, and details of constituent elements. We present a visualization technique that addresses these challenges. The approach uses set intersection graphs that explicitly visualize each set intersection as a rectangular node and elements as circles inside them. We represent the graph as a layered node-link diagram using colors to indicate the sets. The layers reflect different levels of intersections, from the base sets in the lowest layer to potentially the intersection of all sets in the highest layer. We provide different perspectives to show temporal changes in set membership. Graphs for individual, two, and all timesteps are visualized in static, diff, and aggregated views. Together with linked views and filters, the technique supports the detailed exploration of dynamic set data. We demonstrate the effectiveness of the proposed approach by discussing two application examples. The submitted supplemental material contains a video showing proposed interactions in the implementation and the prototype itself.

1. Introduction

Datasets usually contain data items that belong in multiple categories. Modeling categories as sets and data items as elements has helped in the visual analysis of these datasets. A large number of existing static set visualization techniques [AMA*16] stand testimony to the statement. However, we still lack techniques which: (a) model and visualize the details of element memberships in a static set (e.g., element-set membership weight) and (b) represent temporal changes in element memberships. In this paper, we propose an approach that addresses these challenges.

Many real world scenarios can be modeled as dynamic sets. For instance, we can see the business strategy of companies by analyzing the temporal change of their product lines. Figure 1 visualizes a sample dataset where product categories are modeled as sets, while companies are set elements. If a company makes a particular type

† e-mail: shivam.agarwal@paluno.uni-due.de
‡ e-mail: gleb.tkachev@visus.uni-stuttgart.de
§ e-mail: michel.wermelinger@open.ac.uk
¶ e-mail: fabian.beck@paluno.uni-due.de

of product, then it belongs to the corresponding set. The dataset contains information of 13 companies that make products across three categories over three decades (each decade is a timestep).

Our approach centers on a layered set intersection graph, where each node represents a base set or an intersection of base sets, and edges represent direct subset relationships. Sets are shown as rectangles with their elements (circles) inside them. Nodes in layer n represent the intersections of n base sets, e.g., ***Gaming Console*** $\cap$ ***Search Engine*** $\cap$ ***Operating System*** is in layer L_3 in Figure 1.

Representatives of an element appear in multiple nodes if being part of multiple sets and intersections. The *degree of an element* is the number of base sets it belongs to. An element is *exclusive* to a base set if it does not belong to any other set. More generally, an element is exclusive to an intersection if its degree is the number of base sets of the intersection, i.e., if the element does not belong to any set besides those in the intersection. We mark exclusive elements with a hat. These elements do not appear in any layer above.

We compute a set intersection graph for each timestep. The graphs are visualized individually or summarized by an aggregated representation across all timesteps (Figure 1a). A *diff* view can show the exact changes between any two timesteps. For instance, in Figure 1b, *Google* is highlighted, and a tapered directed edge shows that the strategy of the company shifted in the *2010s* from *2000s* to producing both ***Search Engine*** and ***Operating System***. The vertical layout of layers allows us to see the trajectory of individual elements over time, from 'specialists' (belonging to one set) to 'generalists' (belonging to multiple sets) or vice versa. To enable in-depth exploration, we integrate filters for querying, a navigable timeline, a chart showing degree distribution, and an element list.

Our contribution is a novel application-independent visualization technique for analysis of membership details in overlapping sets and temporal changes in set membership (Section 4). To the best of our knowledge, this is the first technique to allow a rich analysis of evolving sets at the level of individual elements. Two application examples demonstrate the effectiveness and generality of the proposed approach (Section 5). We implemented the approach as a Web-based prototype[†]. The supplemental material [Aga20] includes a video demonstration of the system and the prototype itself.

2. Related Work

As surveyed by Alsallakh *et al.* [AMA*16], there exist various set visualization techniques, with different foci and layouts. First, we discuss **static set visualization** techniques that are most relevant to our work, thereby leaving out a comprehensive survey.

Graph-based techniques represent elements and sets as nodes of a bipartite graph connected by edges (to indicate set membership), either with separate areas for element and set nodes [DHRD12; SJUS08; SGL08] or within a shared space [BH11; Mis06]. Analyzing an overlap becomes difficult in both layouts, as it has to be inferred by following the individual edges and finding all element nodes contained in the overlap. *RadialSets* [AAMH13] addresses this by representing only sets and their intersections as nodes. However, elements are aggregated within each node, which hides membership details of elements. Additionally, intersections of over four base sets are not shown in the visualization. *UpSet* [LGS*14] uses a matrix layout where each row represents an overlap. Hence, analyzing a specific overlap becomes easy. However, the technique aggregates the elements and focuses on showing the cardinality and additional attributes of elements in each set and their overlaps.

Aggregating elements of a set significantly increases the scalability. However, it does not allow the analysis of those scenarios, where set elements differ from each other (e.g., when they have different weights of membership in the same set), which is common in real world scenarios. For instance, the importance (weight of membership) of a text keyword (element) in a corpus (set) is encoded via font size in *Parallel Tag Clouds* [CVW09]. However, they do not mark whether the elements are present in additional sets or not (exclusivity). In our approach, we address the challenge of visualizing differences between elements in terms of their set membership weights and their exclusivity.

Formal concept analysis (FCA) [GW12; SKG16] is used to find and group equivalent elements and results in a concept lattice. A concept represents which elements (in the example: companies) are associated with which sets (the type of products manufactured). Subset relations between concepts gives a hierarchical structure to the lattice. These lattices have been visualized in various ways and found to be effective [GEF17; WYS09; EV10; MMLA12; VGRH03; EDB04]. We take inspiration from these techniques and propose a layered graph-based representation.

There exist only a few approaches that have partly addressed the challenge of **visualizing dynamic sets**. *Set Streams* [AB20] represents sets and their non-empty intersections as individual rows while showing the changing memberships of elements as streams in a horizontal layout. However, since elements are aggregated, it does not show details of the membership for each element. *TimeSets* [NXWW16] visualize sets of events on a timeline. An event has a timespan, modeled as an element, and may belong to more than one set. However, the set membership of elements (events) does not change over time. Another technique shows the change in membership of elements but restricts them to belong to only one set in a timestep [vBA*12]. Valdivia *et al.* [VBP*20] show the dynamic changes as a hypergraph—hyperedge connecting several vertices. However, set intersections are not shown explicitly, and they have to be inferred by following a hyperedge. The animation based approach of Mizuno *et al.* [MWTI19] optimizes the sequence of dynamic changes in a node-link based set representation. However, with animation, it becomes difficult to keep track of all the changes happening to multiple elements and sets.

Our approach is more general (an element can be in multiple sets at any time and in different sets over time) and simpler (we show set membership changes without using animation).

3. Set Intersection Graph

To provide an intuitive understanding of our approach, Figure 2 shows its equivalence with a Venn diagram. To explicitly represent all set overlaps, we construct a static set intersection graph for each

[†] Hosted at: https://vis-tools.paluno.uni-due.de/dsv/

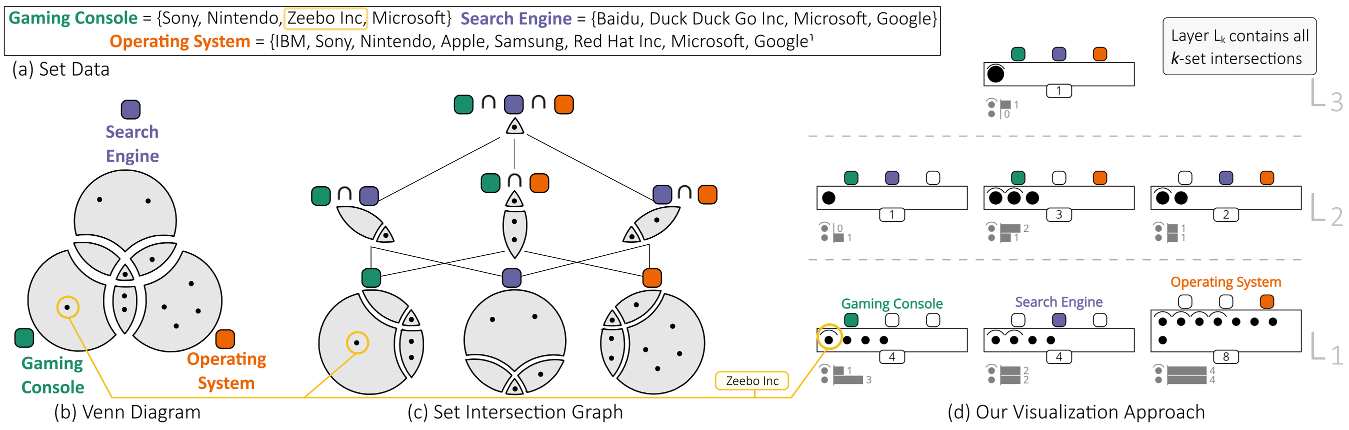

Gaming Console = {Sony, Nintendo, Zeebo Inc, Microsoft} Search Engine = {Baidu, Duck Duck Go Inc, Microsoft, Google}
Operating System = {IBM, Sony, Nintendo, Apple, Samsung, Red Hat Inc, Microsoft, Google}
(a) Set Data
Search Engine
Gaming Console
Operating System
(b) Venn Diagram
Zeebo Inc
(c) Set Intersection Graph
Layer L_k contains all k-set intersections
L3
L2
L1
Gaming Console
Search Engine
Operating System
(d) Our Visualization Approach

Figure 2: *The sample dataset for one timestep (2010s) with 3 types of products as sets (encoded with colors) and companies as elements, based on the type of products they manufacture. (a) The raw data. (b) The data in a Venn diagram. (c) The constructed set intersection graph. (d) Our visualization approach representing layered set intersection graph. Element* Zeebo Inc *is highlighted in all representations. A hat marker (⌒) indicates the exclusive presence of an element–it does not belong to any other base set.*

timestep (Figure 2c). As shown in Figure 2d, we visualize each set and intersection (graph node) by a rectangle and elements of that set as black circles inside the rectangle.

Our set intersection graphs are inspired by concept lattices used in FCA. The top concept of the lattice (topmost node in the graph shown in Figure 2c) has those elements that are present in all sets; the bottom concept has those elements that do not belong to any set. For most realistic datasets, this concept is empty and thus not visualized in Figure 2c. The video in the supplemental material [Aga20] shows an animation of Figure 2 to further help understand the construction and visualization of the layered set intersection graph.

3.1. Data Model

The input for our visualization approach is a non-empty set of m elements $E = \{e_1, e_2, \ldots, e_m\}$ and a family of n base sets $F = \{S_1, S_2, \ldots, S_n\}$ such that $S_i \subseteq E$. Each element can belong to one or more base sets, which undergoes discrete temporal changes. We represent the time dimension as an ordered sequence of p timesteps $T = \langle t_1, t_2, \ldots, t_p \rangle$ $(\forall k < k' : t_k < t_{k'})$. Depending on the application, the timesteps can be interpreted as snapshots or time ranges.

An $m \times n$ matrix W^k contains the data for timestep t_k where rows represent elements ($|E| = m$) and columns represent base sets ($|F| = n$). If cell $w_{ij}^k > 0$, then element e_i is in base set S_j. The value of the cell determines the element's weight of membership in that set. If there is no meaningful weight definition for a certain application, a binary value is sufficient (e.g., $w_{ij}^k \in \{0, 1\}$).

3.2. Set Intersection Graphs

We construct a graph where every node represents a subset of F and contains as elements the intersection of elements of those base sets, e.g., the node for $X = \{S_1, S_3, S_4\}$ contains the elements in $S_1 \cap S_3 \cap S_4$. We include all nodes in the graph that result in non-empty intersections (including a node for each base set, but excluding a node for $\emptyset \subset F$). We add an edge between nodes X_i and X_j if the former is a direct subset of the latter, i.e., $X_i \cup \{S_k\} = X_j$ with $S_k \in F$.

The membership weight of an element e_i in node $X = \{S_j\}$ at timestep t_k is given by w_{ij}^k. For a set intersection, a weight has to be computed. We chose to sum up the element's weights of memberships across all the base sets in X. Formally, the weight of an element e_i in a vertex X at timestep t_k is computed as:

$$W'(e_i, X, t_k) = \sum w_{ij}^k, \quad \forall S_j \in X \qquad \text{(Equation 1)}$$

For instance, if an element is in base sets S_2 and S_4 with weights 1 and 2, it is in $S_2 \cap S_4$ with weight 3. We repeat the process to get *set intersection graphs* $(G^1, G^2, \ldots, G^p)$, one per timestep.

3.3. Aggregated Set Intersection Graph

To compute the fixed layout across all timesteps, we build a *super-graph* [BBDW17] by merging the set intersection graphs of all timesteps. The super-graph nodes are formed by merging equivalent nodes across all set intersection graphs. Nodes from the set intersection graphs for timesteps t_k and $t_{k'}$ are equivalent if they represent the same subset of F, i.e., $X_i^k = X_{i'}^{k'}$. Edges that connect equivalent nodes at different timesteps are merged accordingly:

$$(X_i^k, X_j^k) \equiv (X_{i'}^{k'}, X_{j'}^{k'}) \Leftrightarrow X_i^k \equiv X_{i'}^{k'} \wedge X_j^k \equiv X_{j'}^{k'}$$

Hence, the resulting *super-graph* contains all nodes or edges that are present in at least one timestep, with equivalent nodes and edges contained only once (no replication).

4. Our Visualization Approach

In our approach, we compute a layered layout of set intersection graphs (Section 4.1). The membership of elements in individual timesteps is visualized through static set representation (Section 4.2). An aggregated representation provides an overview of the

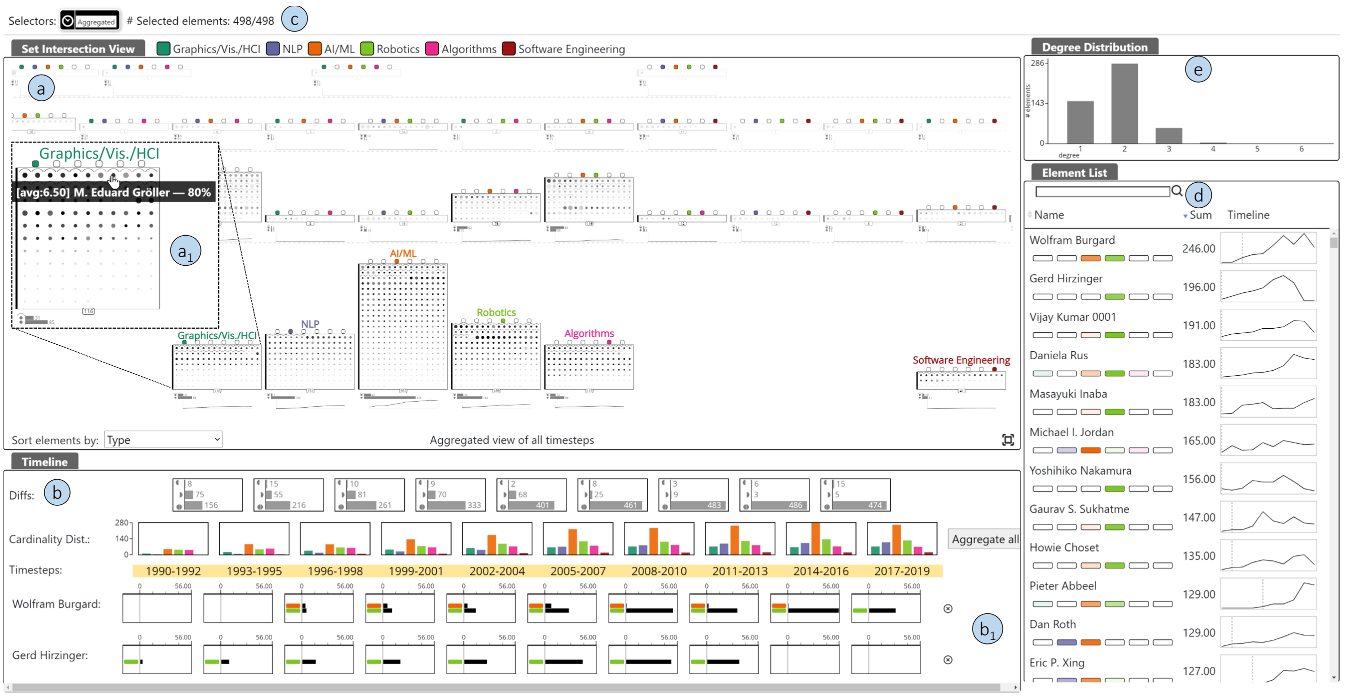


Figure 3: *The interface of the proposed visualization technique with: (a) a set intersection view in the middle (here, showing an aggregated representation), (b) a timeline view with cardinality distribution of base sets in each timestep and relevant statistics for diff representations, (c) applied filters, (d) a list of elements, and (e) degree distribution. The dataset shows research areas in Computer Science as sets (encoded in colors), researchers as set elements, and the number of publications in the respective area as element-set weight. The evolution chart of two selected researchers is shown in* b_1 *for later discussion (Section 5.1).*

temporal changes (Section 4.3). Changes across any two selected timesteps are visualized by a *diff* view (Section 4.4). Interactive filtering and linked views (Section 4.5) enable an in-depth exploration of the set data. Figure 3 shows the prototype's interface.

4.1. Layered Layout of Set Intersection Graphs

To maintain stability while flipping through graphs of different timesteps, equivalent nodes (sets) should retain their position as it helps to maintain a *mental map* of the graphs [BBDW17]. We use aggregated set intersection graphs for a global layout as it contains all nodes and edges that appear in at least one timestep. We compute a fixed spatial position for each node in the graph. This results in a stable and global layout of the graph.

The placement of nodes and links in the aggregated graph follows a layered graph layout based on the Sugiyama algorithm [STT81]. Since a subset relation, as represented by edges, cannot form cycles, the cycle removal step of the Sugiyama algorithm is skipped. Furthermore, we replace the topology-based layer computation by placing the nodes (intersections) in layers based on the number of participating sets in an intersection. Doing so gives semantics to the layers: a layer L_k contains only k-set intersections. For instance, a node representing [***Gaming Console*** ∩ ***Search Engine*** ∩ ***Operating System***] is assigned to L_3 (Figure 2d top). All nodes are ordered within layers according to the barycentric heuristic to minimize edge crossings.

4.2. Static Set Representation

We visualize the nodes of the set intersection graph as rectangles. Small colored boxes identify the participating sets in an intersection (▢ 🟪 🟧 = ***Search Engine*** ∩ ***Operating System***). Elements belonging to an intersection are shown as circles inside the corresponding rectangle. We chose a circular shape because it can be divided into two distinguishable regions (semi-circles) required for the *diff* representation (Section 4.4). The area of a circle encodes the corresponding element's weight of membership (Equation 1).

Exclusive elements are marked with a hat ($\frown$) on top of the corresponding circles, indicating the highest layer in which an element is present (as the element does not have additional memberships). All circles inside a rectangle are ordered by their type (exclusive and non-exclusive). Within each type, elements are sorted by their decreasing membership weight. Such a representation enables us to see the distribution of elements within a set. Other criteria provided include sorting elements by their weight or name (Figure 3a bottom). The height of a node (rectangular box) indicates the cardinality of the corresponding intersection. We show the exact cardinality as a number centered at the bottom of the node. The number of exclusive and non-exclusive elements is visualized by horizontal bars below the node (Figure 3a_1).

4.3. Aggregated Set Representation

To show an overview of temporal changes in set memberships across all timesteps, we provide a time aggregated set representation. We average the weights of each element individually across all timesteps in individual sets and intersections, which is encoded via the area of corresponding circles (Figure 3a_1). We keep the visual representation similar to static sets to minimize the need for memorizing additional visual encodings. The aggregated set representation (Figure 3a_1) uses opacity to encode the percentage of timesteps a certain visual element (i.e., a set, an element) is present; low opacity (gray) encodes a low percentage while high opacity (black) indicates that the visual element is present at all timesteps. For elements, filled circles in the aggregated set have varying opacity. Similarly, for a set, the left edge of the corresponding rectangular box is thickened and filled with the computed opacity level. In Figure 3a$_1$, the rectangle representing ***Graphics/Vis./HCI*** has a black thick left edge, which means the set was present in every timestep (had at least one element in every timestep).

The percentage of timesteps and average weights can be retrieved interactively on demand (on hovering), as shown in Figure 3a$_1$. Elements can gain or lose membership in sets over time. An element's maximum degree over time is marked with a hat marker in the rectangle(s) representing the corresponding set or their intersection(s). For instance, in Figure 3a$_1$, the highlighted circle shows researcher *M. Eduard Gröller* published an average of 6.5 articles per timestep within the field of ***Graphics/Vis./HCI*** in 80% of the timesteps. The hat marker shows that the researcher never published in any other field together with ***Graphics/Vis./HCI*** in the same timestep.

4.4. Diff Representation

Explicitly pointing out differences between two timesteps of a dynamic graph helps in analyzing changes [APP11; RM13; RF14; ZKS11]. Likewise, we propose a *diff view* to represent exact changes in sets between any two timesteps. We divide the visual elements vertically into two parts. The left half shows data of the earlier timestep (t_k) while the right half shows data of the later timestep ($t_{k'}$ where $k < k'$). A circle (element) is divided into two halves (left and right semi-circles), to indicate its presence in two timesteps, as shown in Figure 4a. Similarly, hat markers are also vertically divided into two arcs, showing the maximum degree of an element in the two timesteps. Likewise, the set cardinalities at two timesteps are shown near the bottom left and bottom right corners of the corresponding rectangular box (Figure 4d). After experimenting with several designs, we finally chose to display rectangular boxes in the *diff view* with different stroke patterns to represent their existence only in t_k with a dotted border, only in $t_{k'}$ with a dashed border, and in both timesteps with a continuous border (Figure 4a).

We show membership changes with tapered edges, which are available on-demand to reduce clutter. As shown in Figure 4b, a tapered edge highlights the shift in the element's membership from ***Search Engine*** to ***Operating System***. The tapered edge shown in the figure is horizontal. If the element's degree (number of base sets it belongs to) changes across two timesteps, the tapered edge is drawn between two layers. As shown in Figure 1b, the inter-layer tapered edge shows that *Google* gained membership of the ***Operating System*** set at a later timestep. Summary edges abstract tapered edges with the same source and destination layers and show the number of elements inside a circular base at the origin. For instance, from Figure 4c we can infer that five elements with degree 1 (source is L_1) in t_1 gained membership in two additional sets in t_2 because their degree became 3 (destination is L_3).

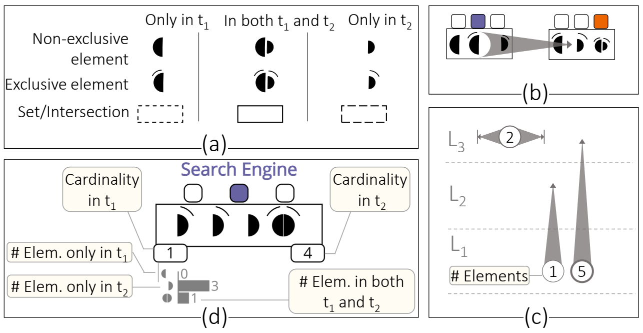


Figure 4: *Diff between t_1 and t_2 showing: (a) visual encodings for elements (circles) and sets (rectangles), (b) change in set membership by tapered edge, (c) group of elements undergoing similar changes by summary edges, and (d) diff view of set **Search Engine** with annotations.*

Three cases arise based on the presence of elements and existence of sets in two timesteps: (i) present only in the earlier timestep (t_k), (ii) only in the later timestep ($t_{k'}$), and (iii) present in both timesteps. These cases are distinguishable through our chosen encodings, as shown in Figure 4a. For each set, the number of elements in the three cases is visualized by horizontal bars beneath the corresponding rectangular boxes (Figure 4d). Inside a rectangular box, the elements are primarily ordered by the three cases, while secondary ordering is on their weight of membership.

4.5. Linked Views, Filters, and Interactions

Besides the *set intersection view* given by the layered graph, we integrate other views to visualize details of set data, provide filters and interactions supporting in-depth visual analysis. The video in supplemental material [Aga20] shows the working of linked views, filters, and interactions in action.

A **Timeline** shows all the labeled timesteps ordered chronologically and drawn in a horizontal layout (Figure 3b). Small rectangles above each timestep label contain colored vertical bars to indicate the cardinality of base sets. The ability to select the timesteps allows easy navigation between different timesteps. Keyboard navigation with arrow keys is also supported. Temporal aggregation can be done by clicking the *'Aggregate all'* button. The selection of two timesteps for *diff view* requires clicking any two timesteps while holding the *Ctrl* key. Additionally, the *diff* view between adjacent timesteps can be retrieved by selecting the rectangles in the *'Diffs'* row above. Each rectangle in the row contains three horizontal bars showing the number of elements present: only in the left timestep, only in the right timestep, and in both timesteps.

An **Evolution Chart** of an element is a series of rectangles in a row placed below the timestep labels (Figure 3b_1). Each rectangle shows an element's weight of membership in sets for one timestep, encoded as horizontal bars. With this representation, the evolution chart enables a comparison between elements. It is drawn on demand when an element is selected (Figure 8).

An **Element List** shows a list of elements as rows (after applying current filters) on the right side of the interface (Figure 3d). Each row in the list represents one element with additional details: name of the element, the sum of its membership weights in all sets among selected timestep(s), and a timeline showing a temporal variation of its cumulative membership weights ( ). The vertical dashed line in the timeline marks the timestep when the element first appeared. In the *static view*, colored boxes in each row indicate the membership of an element in the corresponding sets (▭ ▬ ▭). In the *diff view*, each box is subdivided vertically into two halves (▭ ▬ ▬) and filled according to the presence of the element in the corresponding sets across two timesteps. Whereas in the *aggregated view*, the opacity of color indicates the percentage of timesteps the element is present in the corresponding base set (▭ ▬ ▬). Clicking on a row selects the element, highlights it in the *set intersection view*, and draws its *evolution chart*.

The **Degree Distribution Chart** shows a distribution of filtered elements in terms of the degree of individual elements (Figure 3e). Existing visualizations such as *RadialSets* [AAMH13] have shown the usefulness of degree distributions in the analysis of set data. In the *diff view*, each degree bar is vertically split into two bars, one for each timestep.

Filters: The ability to query and filter a group of elements based on visual selection is a powerful way to analyze sets. We integrate a mechanism where a user can simultaneously select: (i) timesteps, (ii) sets and intersections, (iii) *summary edges*, and (iv) degree of elements. Each selection acts as a filter that returns a group of elements fulfilling the selected criteria. We represent filters with tags (e.g., 2010's , Search Engine ✕ , L 1 to L2 ✕ , and degree=1 ✕) above the *set intersection view* (Figure 3c).

Other Interactions: The *set intersection view* (Figure 3a) supports panning and zooming. Hovering over any element shows connecting edges between the related sets and intersections. The element is selected when its circle or semi-circle is clicked. The selection stays persistent when selecting another timestep or view (*static, aggregate, and diff*). With this feature, one can trace an element across different timesteps when flipping through *diff* views.

5. Application Examples

To demonstrate the applicability and effectiveness of our approach, we study two realistic application examples. The prototype containing these examples is in the supplemental material [Aga20].

5.1. Researchers' Field of Interest

Publication venues (conferences, journals, etc.) can be mapped to fields of science, fields can be modeled as sets, and researchers as set elements. Publication by a researcher in a field of science determines the set membership. The publication year adds the temporal component. We collected publication data from conferences of 6 research fields (sets). We filter those researchers who published at least 30 articles over all fields and timesteps, obtaining 498 researchers. Since researchers can publish in multiple fields, they can appear in multiple sets. Hence, the six sets overlap, with 32 different set intersections. The dataset covers publications during 1990–2019, divided into ten timesteps of three years each.

We first discuss observations derived from the last timestep (2017–2019). Our approach can report findings based on the extent of overlap among sets and their cardinalities. We observe that there is a high overlap among sets (Figure 5a). The set ***AI/ML*** is the largest (height of the rectangle), while two intersections contain only one element. Our approach enables analysis based on the membership weight of elements. Comparing two different set intersections: $X_A = \{$ ***Graphics/Vis./HCI***, ***Algorithms*** $\}$ vs. $X_B = \{$***Graphics/Vis./HCI***, ***NLP***, ***AI/ML***, ***Robotics***$\}$ (Figure 5a_1), we find that both contain only one element. Through different sizes of circles, our approach can highlight the differences in membership weights of elements. X_A has one element *David R. Karger* (small circle), who published three papers, while X_B has a researcher *Sergey Levine* (big circle), who published 48 papers.

Suppose we are interested in the researchers who have published in both ***NLP*** and ***Robotics*** during 2017–2019. In Figure 5a, the rectangular node representing the intersection of the two sets contains five circles. None has a hat marker, indicating that no researcher published in just the two fields. For the base sets ***NLP*** and ***Robotics***, in Figure 5a, we can see that the heights of corresponding rectangles are similar, suggesting that their cardinalities are almost the same (112 and 128, respectively). Our approach enables further analysis based on the circle size and their distribution. On sorting by the membership weight, we see that the size of circles in ***Robotics*** is larger than in ***NLP*** (both exclusive and non-exclusive elements). Hence, we can say that the number of publications by most individual researchers in ***Robotics*** is higher than researchers in ***NLP***. On hovering, we find that the maximum number of publications in ***Robotics*** is 34 by *Masayuki Inaba*, whereas in ***NLP*** it is 25 by *Graham Neubig*.

Next, we show how our technique can visualize the temporal changes in sets from an overview perspective, followed by exploring changes in the membership of specific elements.

A snapshot of the interface with the publication dataset is shown in Figure 3. The aggregated view in the middle (Figure 3a) shows that every set is present in all the timesteps (black left edge of rectangles in the bottom layer). Vertical colored bars (Figure 3b) and the line chart below the nodes in L_1 (Figure 3a) shows that the number of researchers in ***AI/ML*** has grown rapidly compared to other fields. While panning the aggregated view, we observe that there are only four layers in the dataset. It indicates that no researcher has published in more than four research fields in a single timestep.

To find the researchers with the highest number of publications and their field of research, we sort the element list by decreasing order of '*Sum*' column (left-click on the column header), as shown in Figure 3d. To know the researcher's field of study, we look at the colored boxes in each row. The colored boxes (orange and green) for the first researcher *Wolfram Burgard* ( ) shows that he has published only in ***AI/ML*** and ***Robotics***. A pattern can be seen from the first five rows in the list. In each row, the green-colored box

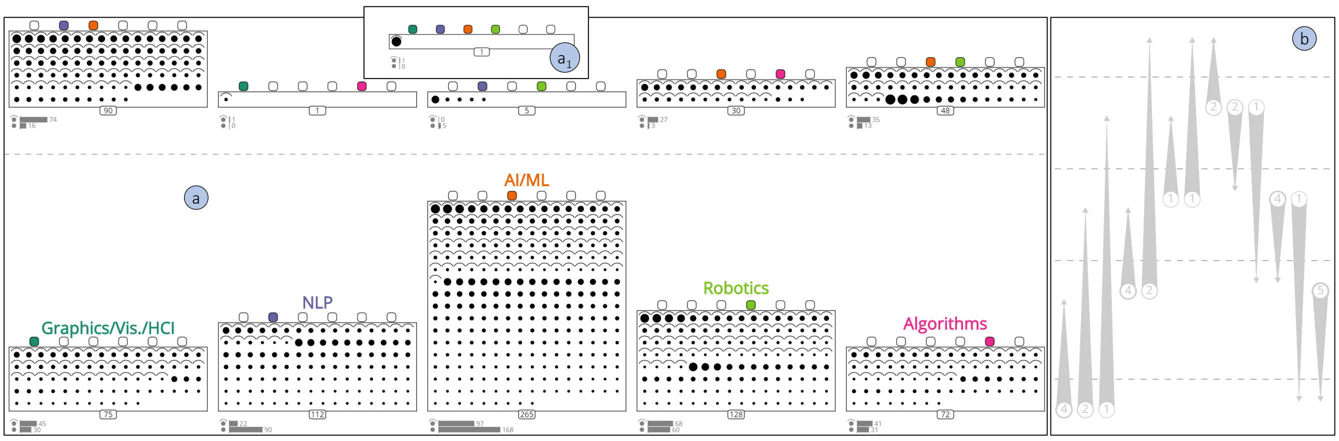


Figure 5: *Snippets of the interface showing (a) set intersection view of the last timestep (2017-2019) from Computer Science research dataset (Section 5.1) and (b) summary edges in the diff view between 2016 and 2017 timesteps from Linux GitHub repository dataset (Section 5.2).*

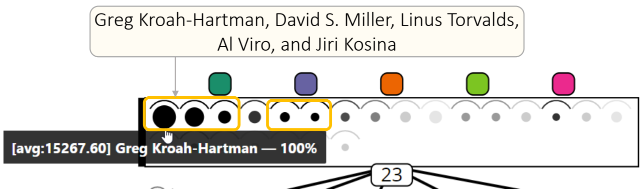

Greg Kroah-Hartman, David S. Miller, Linus Torvalds, Al Viro, and Jiri Kosina
[avg:15267.60] Greg Kroah-Hartman — 100%
23

Figure 6: *Consistent contributors in five modules of Linux GitHub repository across all timesteps.*

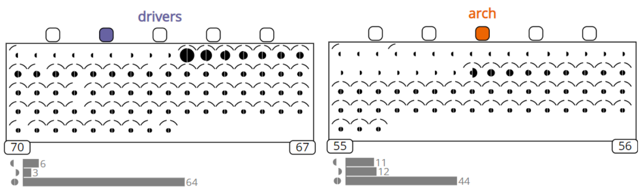


Figure 7: *Comparing stability of developer contributions among modules **drivers** and **arch** in the diff view between 2016 and 2017.*

is very prominent (less transparency), which indicates that the top five researchers have consistently published in ***Robotics***. The second row shows researcher *Gerd Hirzinger* ( ) with only a green colored box, indicating his specialization in ***Robotics***. These observations are confirmed by their evolution chart (Figure 3b_1). As seen from the timelines of the first five rows, the researchers are active except *Gerd Hirzinger*, who stopped publishing in the last two timesteps. Additionally, the number of articles they publish per year has been declining except for *Mayasuki Inaba* ( ). For further exploration, publication details of a researcher are available on right-clicking the corresponding element.

5.2. Evolution of Developer Activities in Software Projects

In this example, we analyze changes in software development activities. Analyses like these can show staff churn, productivity differences, and modules requiring more work, thus helping manage a software project [WYS09]. We study 5 Linux modules from its GitHub repository. The modules (***fs***, ***drivers***, ***arch***, ***net***, and ***kernel***) are the sets, and the elements are the committers (developers). The membership weight is the number of commits done by a developer to a module. We divide the repository evolution from 2008 to 2017 into ten yearly timesteps and filter the developers who made at least 100 commits to these modules, obtaining 111 committers.

We select the aggregate view by clicking the *'Aggregate all'* button. Since presence across all timesteps is encoded via opacity, to find the most consistent developers, we look for black circles. Hovering over them reveals the developer names, average commits in every timestep, and percentage of presence in all timesteps, as shown in Figure 6. These developers contributed to all modules (the rectangle is in layer L_5) in every timestep. The circle sizes show a large difference in the average number of commits, from 15267.60 for *Greg Kroah-Hartman* (Figure 6) to 834.20 for *Jiri Kosina*.

To see the changes in developer activities in the last two timesteps (2016 and 2017), we select the *diff* view between 2016 and 2017. We mention two remarkable changes:

• **Module stability:** In the bottom layer, we focus on the two biggest rectangles representing the ***drivers*** and ***arch*** modules (Figure 7). The ***arch*** module has the least change in terms of cardinality (55 to 56). A change in cardinality alone is not a good indicator of stability. On a closer look, we see that the module has many semi-circles. The horizontal bars beneath the rectangle (***arch***) show that 11 previous committers did not contribute in the later timestep. The cardinality remained stable because 12 new developers contributed to the module. In contrast, we see that the ***drivers*** module has the most significant number of developers (64), who contributed in both timesteps. Hence, across the two timesteps, the ***drivers*** module was the most stable in terms of developer contributions.

• **Developers shifting their focus among modules:** Zooming on the summary edges in the *diff* view, we see many inter-layer tapered edges going up and down (Figure 5b). Upward edges indi-

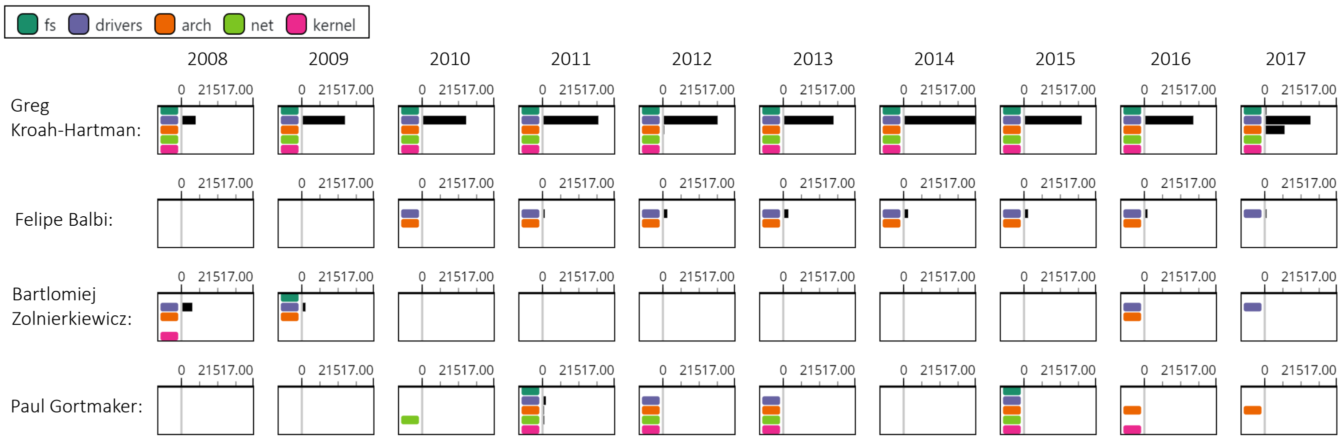


Figure 8: *Evolution charts of four committers, showing different contribution patterns across five Linux modules (encoded in color).*

cate that developers contributed to more modules than before, and vice-versa for downward edges. The summary edge from L_2 to L_1 indicates that five developers narrowed their focus to only one module. Selecting the edge populates the element list with their names.

To highlight the different patterns of contributions among developers, we use evolution charts (Figure 8). We observe stable and consistent contribution patterns to two modules by *Felipe* and to all five modules by *Greg*. We also see an inconsistent contribution across timesteps (*Paul*). Additionally, we observe a developer not contributing to any module for some years (*Bartlomiej*).

6. Discussion

Scalability for dynamic set visualization includes the number of sets and relevant set intersections, the number of elements, the level of detail shown for set membership, and the number of timesteps that are represented. Like any approach that explicitly models set intersections, the number of relevant intersections can explode. In the worst case, the set intersection graph has 2^n nodes. But in practice, often, a large majority is omitted as these correspond to empty set intersections. We observed in the tested data that up to six significantly overlapping sets could be represented without the visualization becoming too dense. Region-based or line-based overlay techniques [AMA*16, Section 4.2] do not scale any better if sets significantly overlap. Some approaches only show intersections of two sets (e.g., *AggreSet* [YEB16]), but this limits the analytical power. With respect to the number of elements, existing aggregation-based set visualization techniques (e.g., *UpSet* [LGS*14]) are more scalable. However, they hide the membership details of an element. Visualization techniques based on aggregating elements are highly scalable, but they are not directly comparable to our technique. We do not aggregate any elements because we want to preserve information on single elements for in-depth analysis. As demonstrated, representing five hundred elements is feasible. Other approaches that show individual elements (e.g., *Bubble Sets* [CPC09], *OnSet* [SMDS14]) scale similarly to our technique with respect to the number of elements. As demonstrated, our approach is scalable up to ten timesteps, which is similar to existing dynamic set visualizations (e.g., *Set Streams* [AB20]). Since the changes in element memberships might not be interesting for all timesteps or for all set intersections, future work would include using data analysis techniques to highlight the most relevant timesteps and intersections.

Unlike most set visualization techniques, our approach can model and visualize the membership weight of an element for each set it belongs to, together with the dynamic associations between elements and sets over time. The proposed technique allows in-depth visual analysis of how sets and their overlaps grow and shrink, and how elements ‘migrate’ through sets. This has many applications. We presented one toy example (Figure 1) and two real datasets. They show how our visualization approach allows exploring over time which products, research fields, and software modules are more or less active. Analysis of the examples reveals which companies or people contribute most, or more consistently, and how they become generalists (move towards the top layer) or stay specialists (in the lower layers). This has practical value in market analysis, supervisor choice, workload planning, team composition, setting a software roadmap, etc. The prototype in the supplementary material [Aga20] has further datasets that show the effectiveness and generality of our approach.

7. Conclusion

We presented a first visualization approach that allows an in-depth exploration of changes to set memberships down to the level of individual elements and their membership weights. The visualization is based on layered set intersection graphs. The layout corresponds to a specialization or generalization hierarchy of elements. Interpreting the graph as a dynamic graph allowed us to show set membership changes with respect to different perspectives.

Acknowledgments

Shivam Agarwal is indebted to Ankita Bajpai for her suggestions on datasets and applications of the technique. This work has been partly funded by Deutsche Forschungsgemeinschaft (DFG) through research grant 288909335.

References

[AAMH13] ALSALLAKH, BILAL, AIGNER, WOLFGANG, MIKSCH, SILVIA, and HAUSER, HELWIG. "Radial Sets: Interactive Visual Analysis of Large Overlapping Sets". *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), 2496–2505. DOI: 10.1109/TVCG.2013.184 2, 6.

[AB20] AGARWAL, SHIVAM and BECK, FABIAN. "Set Streams: Visual Exploration of Dynamic Overlapping Sets". *Computer Graphics Forum* 39.3 (2020), 383–391. ISSN: 1467-8659. DOI: doi:10.1111/cgf.13988 2, 8.

[Aga20] AGARWAL, SHIVAM. *Supplementary Material: Visualizing Sets and Changes in Membership Using Layered Set Intersection Graphs*. 2020. DOI: 10.17605/OSF.IO/3ND9H 2, 3, 5, 6, 8.

[AMA*16] ALSALLAKH, BILAL, MICALLEF, LUANA, AIGNER, WOLFGANG, et al. "The State-of-the-Art of Set Visualization". *Computer Graphics Forum* 35.1 (2016), 234–260. DOI: 10.1111/cgf.12722 1, 2, 8.

[APP11] ARCHAMBAULT, DANIEL, PURCHASE, HELEN C, and PINAUD, BRUNO. "Difference Map Readability for Dynamic Graphs". *Graph Drawing*. 2011, 50–61. ISBN: 978-3-642-18468-0. DOI: 10.1007/978-3-642-18469-7_5 5.

[BBDW17] BECK, FABIAN, BURCH, MICHAEL, DIEHL, STEPHAN, and WEISKOPF, DANIEL. "A Taxonomy and Survey of Dynamic Graph Visualization". *Computer Graphics Forum* 36.1 (2017), 133–159. DOI: 10.1111/cgf.12791 3, 4.

[BH11] BORGATTI, STEPHEN P and HALGIN, DANIEL S. "Analyzing affiliation networks". *The SAGE Handbook of Social Network Analysis*. 2011, 417–433. DOI: 10.4135/9781446294413.n28 2.

[CPC09] COLLINS, CHRISTOPHER, PENN, GERALD, and CARPENDALE, SHEELAGH. "Bubble Sets: Revealing set relations with isocontours over existing visualizations". *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), 1009–1016. DOI: 10.1109/TVCG.2009.122 8.

[CVW09] COLLINS, C., VIEGAS, F. B., and WATTENBERG, M. "Parallel Tag Clouds to explore and analyze faceted text corpora". *Proceedings of IEEE Symposium on Visual Analytics Science and Technology*. 2009, 91–98. DOI: 10.1109/VAST.2009.5333443 2.

[DHRD12] DÖRK, MARIAN, HENRY RICHE, NATHALIE, RAMOS, GONZALO, and DUMAIS, SUSAN. "PivotPaths: Strolling through faceted information spaces". *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), 2709–2718. DOI: 10.1109/TVCG.2012.252 2.

[EDB04] EKLUND, PETER, DUCROU, JON, and BRAWN, PETER. "Concept Lattices for Information Visualization: Can Novices Read Line-Diagrams?": *Concept Lattices*. 2004, 57–73. ISBN: 978-3-540-24651-0. DOI: 10.1007/978-3-540-24651-0_7 2.

[EV10] EKLUND, PETER and VILLERD, JEAN. "A Survey of Hybrid Representations of Concept Lattices in Conceptual Knowledge Processing". *Formal Concept Analysis*. 2010, 296–311. ISBN: 978-3-642-11928-6. DOI: 10.1007/978-3-642-11928-6_21 2.

[GEF17] GREENE, GILLIAN J, ESTERHUIZEN, MARVIN, and FISCHER, BERND. "Visualizing and exploring software version control repositories using interactive tag clouds over formal concept lattices". *Information and Software Technology* 87 (2017), 223–241. DOI: 10.1016/j.infsof.2016.12.001 2.

[GW12] GANTER, BERNHARD and WILLE, RUDOLF. *Formal Concept Analysis: Mathematical foundations*. 1st. 2012. DOI: 10.1007/978-3-642-59830-2 2.

[LGS*14] LEX, ALEXANDER, GEHLENBORG, NILS, STROBELT, HENDRIK, et al. "UpSet: Visualization of intersecting sets". *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), 1983–1992. DOI: 10.1109/TVCG.2014.2346248 2, 8.

[Mis06] MISUE, KAZUO. "Drawing bipartite graphs as anchored maps". *Proceedings of Asia-Pacific Symposium on Information Visualisation*. 2006, 169–177. URL: https://dl.acm.org/doi/10.5555/1151903.1151929 2.

[MMLA12] MELO, C., MIKHEEV, A., LE-GRAND, B., and AUFAURE, M. "Cubix: A Visual Analytics Tool for Conceptual and Semantic Data". *Proceedings of International Conference on Data Mining Workshops*. 2012, 894–897. DOI: 10.1109/ICDMW.2012.41 2.

[MWTI19] MIZUNO, KAZUYO, WU, HSIANG-YUN, TAKAHASHI, SHIGEO, and IGARASHI, TAKEO. "Optimizing Stepwise Animation in Dynamic Set Diagrams". *Computer Graphics Forum* 38.3 (2019), 13–24. DOI: 10.1111/cgf.13668 2.

[NXWW16] NGUYEN, PHONG H, XU, KAI, WALKER, RICK, and WONG, BL WILLIAM. "TimeSets: Timeline visualization with set relations". *Information Visualization* 15.3 (2016), 253–269. DOI: 10.1177/1473871615605347 2.

[RF14] RUFIANGE, SÉBASTIEN and FUHRMAN, CHRISTOPHER P. "Visualizing protected variations in evolving software designs". *Journal of Systems and Software* 88 (2014), 231–249. DOI: 10.1016/j.jss.2013.10.044 5.

[RM13] RUFIANGE, SÉBASTIEN and MCGUFFIN, MICHAEL J. "DiffAni: Visualizing Dynamic Graphs with a Hybrid of Difference Maps and Animation". *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), 2556–2565. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.149 5.

[SGL08] STASKO, JOHN, GÖRG, CARSTEN, and LIU, ZHICHENG. "Jigsaw: Supporting investigative analysis through interactive visualization". *Information Visualization* 7.2 (2008), 118–132. DOI: 10.1057/palgrave.ivs.9500180 2.

[SJUS08] SCHULZ, HANS-JÖRG, JOHN, MATHIAS, UNGER, ANDREA, and SCHUMANN, HEIDRUN. "Visual Analysis of Bipartite Biological Networks". *Proceedings of Eurographics Workshop on Visual Computing for Biomedicine*. 2008, 135–142. ISBN: 978-3-905674-13-2. DOI: 10.2312/vcbm/vcbm08/135-142 2.

[SKG16] SINGH, PREM KUMAR, KUMAR, CHERUKURI ASWANI, and GANI, ABDULLAH. "A comprehensive survey on formal concept analysis, its research trends and applications". *International Journal of Applied Mathematics and Computer Science* 26.2 (2016), 495–516. URL: http://eudml.org/doc/280124 2.

[SMDS14] SADANA, RAMIK, MAJOR, TIMOTHY, DOVE, ALISTAIR, and STASKO, JOHN. "OnSet: A Visualization Technique for Large-scale Binary Set Data". *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), 1993–2002. DOI: 10.1109/TVCG.2014.2346249 8.

[STT81] SUGIYAMA, KOZO, TAGAWA, SHOJIRO, and TODA, MITSUHIKO. "Methods for Visual Understanding of Hierarchical System Structures". *IEEE Transactions on Systems, Man, and Cybernetics* 11.2 (1981), 109–125. ISSN: 0018-9472. DOI: 10.1109/tsmc.1981.4308636 4.

[vBA*12] VON LANDESBERGER, T., BREMM, S., ANDRIENKO, N., et al. "Visual analytics methods for categoric spatio-temporal data". *Proceedings of IEEE Conference on Visual Analytics Science and Technology*. 2012, 183–192. DOI: 10.1109/VAST.2012.6400553 2.

[VBP*20] VALDIVIA, PAOLA, BUONO, PAOLO, PLAISANT, CATHERINE, et al. "Analyzing Dynamic Hypergraphs with Parallel Aggregated Ordered Hypergraph Visualization". *IEEE Transactions on Visualization and Computer Graphics (To appear)* (2020). DOI: 10.1109/TVCG.2019.2933196 2.

[VGRH03] VALTCHEV, PETKO, GROSSER, DAVID, ROUME, CYRIL, and HACENE, MOHAMED ROUANE. "Galicia: An Open Platform for Lattices". *Proceedings of International Conference on Conceptual Structures*. 2003, 241–254 2.

[WYS09] WERMELINGER, MICHEL, YU, YIJUN, and STROHMAIER, MARKUS. "Using formal concept analysis to construct and visualise hierarchies of socio-technical relations". *31st International Conference on Software Engineering – Companion Volume*. 2009, 327–330. DOI: 10.1109/ICSE-COMPANION.2009.5071013 2, 7.

[YEB16] YALCIN, M ADIL, ELMQVIST, NIKLAS, and BEDERSON, BENJAMIN B. "AggreSet: Rich and scalable set exploration using visualizations of element aggregations". *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), 688–697. DOI: 10.1109/TVCG.2015.2467051 8.

[ZKS11] ZAMAN, LOUTFOUZ, KALRA, ASHISH, and STUERZLINGER, WOLFGANG. "The Effect of Animation, Dual View, Difference Layers, and Relative Re-Layout in Hierarchical Diagram Differencing". *Proceedings of Graphics Interface*. 2011, 183–190. DOI: 10.5555/1992917.1992947 5.