DOI: 10.1111/j.1467-8659.2012.03080.x
Eurographics Conference on Visualization (EuroVis) 2012
S. Bruckner, S. Miksch, and H. Pfister
(Guest Editors)

Volume 31 (2012), Number 3

Kelp Diagrams: Point Set Membership Visualization

Kasper Dinkla[1], Marc J. van Kreveld[2], Bettina Speckmann[1], and Michel A. Westenberg[1]

[1]Eindhoven University of Technology, The Netherlands.
[2]Utrecht University, The Netherlands.

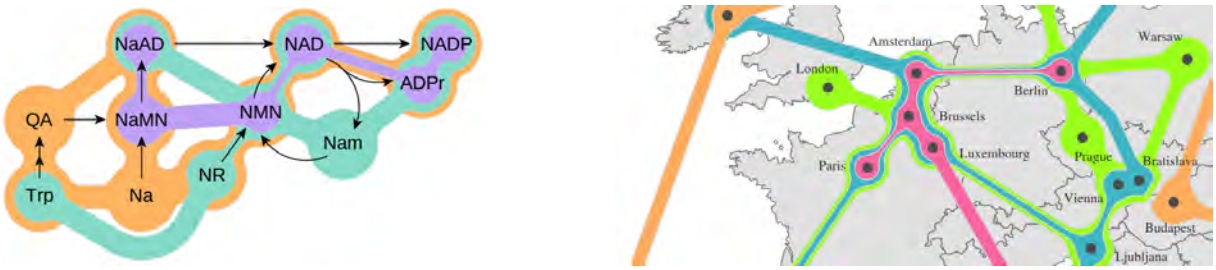


Figure 1: *Kelp Diagrams applied to a metabolic network (left) and cities on a map (right).*

Abstract

We present Kelp Diagrams, a novel method to depict set relations over points, i.e., elements with predefined positions. Our method creates schematic drawings and has been designed to take aesthetic quality, efficiency, and effectiveness into account. This is achieved by a routing algorithm, which links elements that are part of the same set by constructing minimum cost paths over a tangent visibility graph. There are two styles of Kelp Diagrams to depict overlapping sets, a nested and a striped style, each with its own strengths and weaknesses. We compare Kelp Diagrams with two existing methods and show that our approach provides a more consistent and clear depiction of both element locations and their set relations.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Visualization of one or multiple sets, possibly sharing several elements, is a recurrent theme both inside and outside the visualization community. Sometimes, depiction of the sets is the main concern, excluding any other information of contained elements besides some means of identification, i.e., a label. Here, simple visualizations suffice for a small number of sets, such as a table with a mapping of elements to rows and sets to columns, or a more space-efficient Euler diagram. Not every situation warrants such an approach. Sometimes, other data aspects (partially) dictate the positions of the elements' visual representations. For example, geographic places are often best positioned at their real-world coordinates because this is consistent with the existing knowledge of the observer and improves visual orientation. Furthermore, it allows spatial patterns to emerge.

The two state of the art approaches, Bubble Sets [CPC09] and LineSets [ARRC11], use colored shapes to visually connect elements that belong to the same set. Bubble Sets derives an element density function for each set. Isolines are extracted from each function to form shapes around the elements. These shapes are then connected further with links, routed along the elements. LineSets draws a thick colored curve through the elements of each set, making sure that the traversed path over the elements is relatively short. Both approaches generate visualizations that often appear complex and sometimes even convey invalid set memberships. In contrast our method strictly controls where shapes of sets are placed. It consists of three phases (see Fig. 2): the allocation of space around each element such that there is enough room to depict its containing sets; the allocation of space for connecting shapes between demarcation zones with a rout-

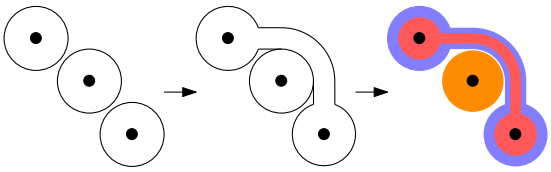


Figure 2: *The three phases of generating Kelp Diagrams.*

ing algorithm; and the generation of actual visualizations, by using the allocated space, in two distinct ways.

In summary, our contribution is:

- Two styles of diagram that emphasize different aspects of set memberships and overlap for elements with a predefined position;
- a routing algorithm for linking elements in a set to support the generation of such diagrams, where aesthetic quality, efficiency, and effectiveness are taken into account.

2. Related Work

Conveying information about multiple sets is a longstanding problem and exists in various forms. When the location of the elements are not specified, then the input is simply a *set system*. Each set can also by interpreted as a *hyperedge* of a *hypergraph* defined on the elements (the vertices of the hypergraph). There are several papers from the graph drawing community [JP87, DBETT99] that discuss how to draw hypergraphs. Most recent efforts have focused on so called *planar supports* for hypergraphs and the associated *subdivision drawings* [KvKS09] (see also [BCPS10a, BCPS10b, BvKM*10] for further theoretical results on specific types of planar supports). Unfortunately most hypergraphs do not have planar supports and hence subdivision drawings are of very limited practical use.

When only sets (or hyperedges) are of concern, the depiction possibilities are numerous. Euler diagrams are well-known and use contours to denote areas that represent sets, which is sometimes referred to as the *subset standard*. Such diagrams can be generated automatically; by abstracting from individual elements [FRM03], or by including representations of the elements and related information [BE01, SAA09, HRD10]. Here, the positions of elements either do not matter as no elements are displayed, or are assigned to optimize the visualization of the sets. The number of elements and sets influences the effectiveness of a visualization. For many elements and groups, augmenting a contour-based visualization is a possibility [ST10], or a highly-interactive analysis environment is a necessity [FMH08].

When dealing with predefined positions of elements, displaying the sets becomes more difficult. If contours are used, like Euler diagrams, a fair division of display space is an issue [BT06, BT09, CPC09]. Another option is to connect highlighted elements with (colored) links [SA06, CC07, SLK*09]. This is referred to as *visual linking*, which can take on sophisticated forms [SWS*11]. However, visual linking focuses on relating elements, not the comparison of sets in a spatial setting. For both approaches the way in which contours or links are placed affects the depiction of the sets but also of the predefined visualization.

3. Problem Analysis

An input problem instance consists of three aspects: positioned elements, the sets that contain them, and the predefined visualization that embeds the elements. We want to depict these sets in combination with the predefined visualization. To determine what makes a good set depiction we enumerate tasks that an observer may wish to perform by interpreting the visualization and hence the data. These tasks impose constraints that any visualization has to fulfill whenever possible, but also provide (conflicting) optimization criteria to improve task performance of the observer.

Supported tasks. The composition of multiple elements and sets—in a spatial setting—brings forth many questions of a comparative nature: To what extent do sets overlap or differ? How close to each other are the elements of a set? Is the containment of an element in a set correlated to its position? We have compiled a list of primitive tasks that have to be supported by a visualization such that an observer is able to answer these questions, or which the observer can perform ad-hoc to gain insight about the data:

T1a determine the position of an element
T1b find an element by position (relative to landmarks)
T1c estimate the density of elements in an area

T2a determine which sets contain a specific element
T2b find the elements that belong to a specific set
T2c estimate the spatial distribution of a specific set

T3 compose a (mental) set from existing sets with operations union, intersect and complement, and apply T1 – T2

The tasks can be composed to answer more complex questions. When dealing with cities on a map, with a set of *large* cities and a set of *industrial* cities, the following questions may be asked: Which cities are large but not industrial? (T2b and T3); Are industrial cities clustered together? (T2c); Is *New York* considered large and/or industrial, and which neighboring cities are similar? (T1a, T2a, T3, and T2b). As shown, answering common questions involves the combination and execution of these primitive tasks. Thus, improving the efficiency at which tasks can be performed, improves the ease at which complex questions can be answered.

Constraints. The visualization has to satisfy the following constraints to enable T1 and T2:

C1 every element is clearly represented in the final visualization at its predefined position (for T1)

C2 every element is clearly marked or contained by a representation of every set that it is a part of (for T2)

These constraints are satisfiable, provided that all elements are visually distinct, i.e., all elements are positioned at a discernible distance from each other. We assume this to be the case because any predefined visualization has to support T1 to be of practical purpose and should therefore have visually distinct elements in the first place.

Aesthetic criteria. The aforementioned constraints guarantee that all tasks can be performed but do not provide direction towards an efficient visualization that allows (composite) tasks to be performed by an observer with little effort. Generation of aesthetic shapes has been a subject of research before for Euler diagrams [SAA09], and the generation of effective graph representations, by reduction of intersections for example, is a common theme in graph drawing [DBETT99]. This has inspired us to list important properties that make for good shapes that depict sets.

Shapes should have low cognitive load:

A1a small area
A1b few and shallow bends
A1c few outline intersections

Not only do aesthetic shapes appeal to the observer, they in general are accompanied by a low cognitive load, i.e., it takes less effort for the observer to process shapes and thus derive the information they are meant to convey. For a set depiction, the faster the shapes that convey sets are interpreted, the faster T2 can be performed.

A small area (A1a) implies less surface to inspect and process, thus less cognitive load. Few and shallow bends (A1b) imply smooth outlines that are easy to distinguish from their surroundings (the predefined visualization). It also means that outlines are short and therefore easier to process. Intersections of outlines (A1c) make them harder to tell apart and discern the area they contain and the elements therein.

Shapes should be effective:

A2a large area
A2b large distance between outlines
A2c little overlap of shapes that depict different sets
A2d strong continuation of shapes that depict the same set

Shapes of large surface area (A2a) attract attention, making the presence of a set explicit (T2). If outlines of shapes are close to each other they are harder to tell apart, often causing the overall shapes to be less pronounced. Likewise, overlapping shapes (A2c) may obfuscate each other and the information they should convey. Both impact T2 negatively.

For T2b—finding the elements that belong to a specific set—to be performed efficiently, the elements have to be associated as a group by the observer. Otherwise, the observer has to scan all elements in a linear fashion to determine their corresponding sets. Incorporating distinguishing features such as color for the shapes is an effective approach. However, creation of a visual continuation of shapes (A2d) causes an even stronger grouping effect (see Fig. 3), on which existing approaches rely as well [CPC09, ARRC11, SWS*11]. In certain situations, like the one shown in Fig. 3(c), strong continuation also emphasizes spatial patterns of elements and sets. This includes improving the detection of spatial clusters (T2c). As is the case for other criteria, continuation is not a strict constraint, i.e., elements that belong to the same set do not have to be connected.

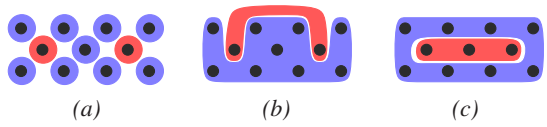


Figure 3: *The added benefit of linking: (a) Elements are associated solely by the colored shapes that contain them; (b) Elements are associated both by color and a common shape; (c) Spatial patterns are emphasized.*

Shapes should not distort element position and density:

A3a little obfuscation of the predefined visualization
A3b strong correspondence between the presence and size of a set's shape, and the presence and density of elements that belong to this set, in an area

Not only do the shapes affect the way in which the predefined visualization is perceived through partial occlusion or obfuscation, they also affect the way in which the elements and their locations are perceived (A3a and A3b). For example, when the set containment of an element is depicted with a large colored circle, this circle will form a stronger visual cue to the presence of an element than the element's own depiction (see Fig. 4(a)). However, this can also affect the perceived position of the element (see Fig. 4(b)) and the density of elements in an area (see Fig. 4(c)).

Many of the stated criteria are in conflict with each other: A2a with A2c, A1a with A1b and A1c because of the routing that is required, A1c with A2d, and A3b with all other criteria. Defining an optimal visualization therefore not only re-

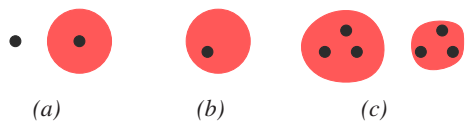


Figure 4: *The distorting effect of shapes on element depiction: (a) An element contained in a shape attracts more attention; (b) The expected position of the element lies at the center of the circle, which conflicts with its actual position; (c) Both sets of elements have the same number of elements, but the difference in size of the shapes suggests otherwise.*

quires quantifying the individual criteria, it also requires the criteria to be prioritized and combined into an overall definition of an optimum. Such an approach has been used for the creation of aesthetically pleasing Euler diagrams [FRM03] and label placement on maps [vDvKSW02], where different criteria are weighted and then used as a fitness function. It underlines the varying expectations that different observers may have of visualizations.

Moreover, when criteria like A1c, which require routing, are included in an overall optimization scheme, the combinatorial complexity of the problem greatly increases and forces the use of heuristic algorithms.

4. Approach

Our approach consists of three phases: allocation of element space, allocation of additional link space, and the generation of visualizations (see Fig. 2). The following pseudo-code provides an overview of the approach, where Sections 4.1, 4.2, and 4.3 elaborate on lines 1–2, 3–10, and 11, respectively. Here, the elements are denoted as $\mathbb{E}$, the predefined element positions as $p(e)$ for $e \in \mathbb{E}$, and the collection of sets as $\mathbb{S}$, where for every $S \in \mathbb{S}$ we have $S \subseteq \mathbb{E}$.

Algorithm *Kelp*$(\mathbb{E}, \mathbb{S})$
1. Derive Voronoi diagram of $\{p(e)|e \in \mathbb{E}\}$.
2. For every $e \in \mathbb{E}$ derive $A(e)$ as the intersection of its Voronoi face and a circle of radius r_e, centered at $p(e)$.
3. Derive embedded graph G_A from $\{A(e)|e \in \mathbb{E}\}$.
4. Derive tangent graph G_T from G_A.
5. **while** best-to-place link l between $p,q \in S \mid S \in \mathbb{S}$ has benefit $b(p,q) > b_t$
6. **do** Add edges of l in G_T to subgraph $G_L(S)$.
7. Derive all-pair shortest paths of $G_L(S)$.
8. Update G_T with intersections introduced by l.
9. Derive all-pair shortest paths of G_T for all $R \in \mathbb{S}$.
10. Derive next best-to-place link from shortest paths in $G_L(R)$ and G_T for all $R \in \mathbb{S}$.
11. Derive visualization style from all $G_L(S) \mid S \in \mathbb{S}$.

4.1. Element space (phase 1)

T2 requires that for each element it is clear to which sets it belongs, hence each element should have ample (and at least equally divided) surrounding space to display its sets. Taking the trade-off between A2a and A2c into account, we want the observer to have a level of control on how much space is used for the set visualization. Given such fixed area per element and considering A1a and A1b, the natural choice of area to allocate around an element is a circle of radius r_e.

It is not always possible to allocate a perfect circle around each element. Sometimes the distance between two elements is smaller than $2r_e$, causing the circles to overlap, or smaller than r_e, causing circles to overlap each other's elements. To resolve this space contention, we first calculate the Voronoi diagram of $\{p(e)|e \in \mathbb{E}\}$ and then, for every element $e \in \mathbb{E}$, intersect the allocated circle of e with the Voronoi face that contains $p(e)$ and use it as the new allocated space $A(e)$ (see Fig. 5). Hence, no allocated space intersects and elements get a fair share of space.

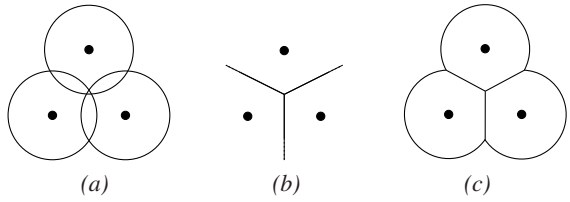


Figure 5: *Allocation of space for three elements: (a) Overlapping circles, centered over elements; (b) The Voronoi faces of the elements' positions; (c) Intersection of circles and Voronoi faces.*

The resulting space partition is stored as an embedded graph $G_A = (V_A, E_A)$, with vertices $V_{\mathbb{E}}$ for the elements, and vertices V_I for the intersection points between Voronoi faces and circles, including the points that are shared by more than two Voronoi faces and lie within an element circle. The edges between these intersection points are either straight (part of a Voronoi face) or circular.

The allocated space of each element is referred to as a *fair* share—not an equal share—because sometimes elements do not receive space that is equal in surface area to their neighboring elements (Fig. 6(a)). This unequal allocation could in certain cases affect the depiction of element density (A3b) negatively. Applying a repulsive force between the elements' circles, and shifting their positions accordingly, would in many situations result in an equal division (Fig. 6(b)), but is not applicable to all situations (Fig. 6(c)). Moreover, manipulating the position or shape of the circles will almost always harm the ability of an observer to find elements by position (T1a) and to determine element density in an area (A3b), see also Fig. 4. The intersection of an element's Voronoi face and circle is simple and allows for visual reinforcement of the element's position, which is discussed in Section 5.

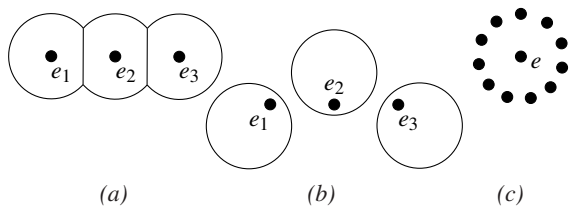


Figure 6: *Area division between elements: (a) Our method allocates an unequal share of space; (b) Possible outcome of a force-based relocation of the circles; (c) Instance where allocation of equal space for element e is not possible without a more complex shape.*

4.2. Link space (phase 2)

A2d states that we should visually connect or link those elements that belong to the same set such that elements from the same set are more easily associated with each other. However, any additional link will result in a more complex visualization, negatively affecting most other criteria. Therefore, we have to allocate link space for every $S \in \mathbb{S}$, such that the advantages of the links in the final visualization outweigh their disadvantages, or cost, as much as possible.

We first consider the *cost* $c(l)$ of placing a link l between $p,q \in S$, where $S \in \mathbb{S}$, in the scene. It can be modeled in a simple way, where $c(l) = c_d d(l) + c_\alpha \alpha(l) + c_I I(l)$. Here, $d(l)$ is the distance covered by l; $\alpha(l)$ is the aggregate angular change of l in radians; $I(l)$ is the number of intersections between the contours of l and the contours of links already placed in the scene; and c_d, c_α, and c_I are weight parameters. Distance is penalized in accordance with A1a, aggregate angular change with A1b, and intersections with A1c.

No intersections should exist between l and any allocated space $A(e)$ where $e \in \mathbb{E} \backslash \{p,q\}$, as l has no right to use space of $A(e)$. Also, $c(l)$ dictates that l runs directly adjacent to any $A(e)$ that it passes, because tightening l, without altering its topology w.r.t. $A(e)$, will always lower $d(l)$ and $\alpha(l)$ without changing $I(l)$ (see Fig. 7).

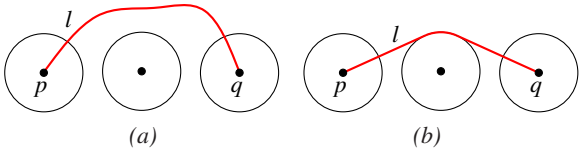


Figure 7: *Two possible paths of a link l between $p,q \in \mathbb{E}$ (in red): (a) l with unnecessarily high cost; (b) l with identical topology but minimized cost.*

As can be seen in Fig. 7, this tightness means that any desirable link can be constructed from the edges in G_A when certain (tangent) edges are added to it, similar to robot motion planning with tangent visibility graphs [WvdBH07]. So we extend G_A to form $G_T = (V_T, E_T)$, which includes (tangent) edges between $A(p)$ and $A(q)$, p and $A(q)$, $A(p)$ and q, and p and q. In addition, for every $v \in V_I$, we have edges to every $p \in \mathbb{E}$ and tangent edges to $A(p)$ (see Fig. 8). This also adds vertices at the tangent points, which split up the original circular edges.

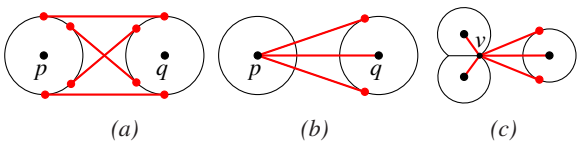


Figure 8: *Examples of the additional edges in E_T (in red): (a) Tangent edges between $A(p)$ and $A(q)$, $p,q \in \mathbb{E}$; (b) Edges from p to q and $A(q)$; (c) Edges from $v \in V_I$.*

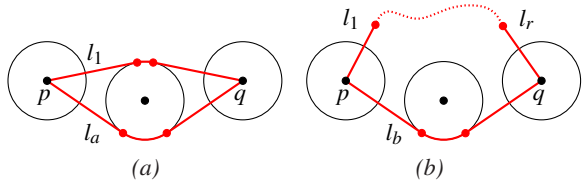


Figure 9: *Benefit of placing a link between $p,q \in S, S \in \mathbb{S}$ is dependent on already placed links: (a) The benefit of placing l_a is low because already placed l_1 has low cost; (b) The benefit of placing l_b is high because already placed chain of links $l_1, l_2, ..., l_r$ has high cost.*

Now, any link l of set $S \in \mathbb{S}$ can be decomposed into a sequence of touching edges $\{e_1, e_2, ..., e_r\} \subseteq E_T$, and $c(l)$ can then be derived from G_T by summing the costs of the individual edges. Based on these costs, it is possible to compute a minimum cost path between $p,q \in \mathbb{E}$ in G_T and thus get l.

Using only cost as a criterion for placing links is not enough when we want to connect elements beyond a spanning tree. When spanning trees have been established for all sets, additional low-cost links are not always of great benefit to the visualization when they are placed. Consider the situations depicted in Fig. 9. For Fig. 9(a) the benefit of placing link l_a is low because a low cost link l_1 is already in place to provide ample visual linking between the elements. For Fig. 9(b) the benefit of placing link l_b is high because the links that already connect p and q sum to a high cost, which means that in the current situation it is hard for the observer to follow links from p to q, while placement of l_b would make this task considerably easier.

Let $G_L(S)$, for every $S \in \mathbb{S}$, be a subgraph of G_T, which contains only edges of links that have so far been added for S. We define the *benefit* of placing a lowest cost link l between $p,q \in S$, as $b(p,q) = \frac{d(p,q)}{c(l)}$, where $d(p,q)$ is the minimum path distance between p and q in $G_L(S)$. Hence, the benefit is higher when the cost of placing l is lower or the smallest distance via already placed links is higher. When p and q are not connected in $G_L(S)$, then $d(p,q) = \infty$. In case the benefit of links has to be compared for disconnected vertices of $G_L(S)$, we compare only by link cost.

Given these definitions, the algorithm places links in a greedy manner: Determine $S \in \mathbb{S}$ and $p,q \in S$ with highest $b(p,q)$. If $b(p,q) > b_t$, where b_t is a benefit threshold parameter, add link l to $G_L(S)$ and repeat the algorithm. When $b(p,q) \leq b_t$, the algorithm terminates.

In our approach so far, routed links have zero width. However, routing links of parameterized radius r_l (width $2r_l$) is achieved by increasing element space allocation from r_e to $r_e + r_l$ (see Fig. 10).

When a link is placed, intersection information is updated for every edge $e \in E_T$, such that we know the number of

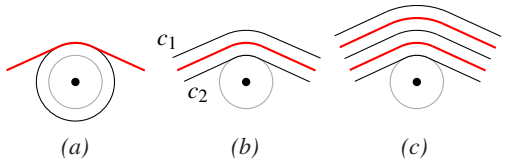


Figure 10: *Link radius: (a) Increase of element space allocation by r_l; (b) Link with allocated space of radius r_l and its contours c_1 and c_2; (c) Two links routed beside each other.*

intersections of e's contours (dilation of e by r_l) with contours of already placed links, i.e., dilation of e_l by r_l for all e_l of $G_L(S), S \in \mathbb{S}$. Tracking intersections between just the edges is not sound as it is possible for edges to be placed next to each other within $2r_l$. Links routed over such edges would therefore intersect without receiving the right intersection penalty. To accommodate routing multiple links adjacent to each other and around the same element space (see Fig. 10(c)), we also extend G_T to include $A(e)$, and corresponding (tangent) edges, dilated with steps of $2r_l$.

Overlapping contours are not counted as intersections. This allows two links l_1 and l_2 to share part of the same path with few intersections $I(l_1)$ and $I(l_2)$. Low-cost sharing of paths is exactly what we want to properly convey overlapping sets, as explained in Section 5.

4.3. Visual styles (phase 3)

The space allocated for elements and their sets' visual links can be used in various ways. We devised two very different diagram styles: nested and striped Kelp.

Nested style. Nested Kelp surrounds elements of every set with a colored shape (see Fig. 11 (top)), where the shapes of every set are stacked on top of each other. It relies on the ability of the observer to mentally distinguish and complete shapes that are partially overlapped by other shapes. The allocated space for elements and links provides a framework to define the diagram such that every shape is sufficiently visible, in correspondence with constraint C2.

We construct the set of shapes $N(S)$ for every $S \in \mathbb{S}$ as follows: For every element $e \in \mathbb{E}$, assume $S_1, S_2, ..., S_r$ contain e, ordered such that $|S_i| \leq |S_j|$ where $i < j$. For every S_i, scale the allocated space $A(e)$ around its centroid by a factor $(\frac{i}{r})^{s_e}$, where s_e is a parameter, and merge it into $N(S_i)$.

For every edge $e \in E_L$, let Q be the set of edges reachable from e in the union of all $G_L(S), S \in \mathbb{S}$, without passing beyond an element node. Assume that $S_1, S_2, ..., S_r$ contain an edge in Q, ordered such that $|S_i| \leq |S_j|$ where $i < j$. Then, for every S_i, dilate e by $r_l(\frac{i}{r})^{s_l}$, where s_l is a parameter, and merge it into $N(S_i)$. Here *dilate* and *erode* are the equivalents of *Minkowski sum* and *Minkowski subtraction* with a circle of a specified radius [dBCvKO08]. Thus, for every element, set membership depictions are bound by the space that was allocated for the element. In addition, depictions are scaled to nest. Links and element circles thus do not become visually dominant when many sets share an element or path.

Links of a set are scaled to nest with links of other sets that partially share a path in G_L. The order in which sets are nested is based on the size of sets, i.e., smaller sets nest in larger sets, which is consistent throughout the diagram. The scaling of both element circles and links, by their level of nesting, depends on parameters s_e and s_l, respectively, such that the share of space allocated to every set can be adjusted to compensate for the effects of perceptual scaling [DTH08].

The shapes $N(S)$, for $S \in \mathbb{S}$, are drawn on top of each other, ordered by $|S|$, where $N(S)$ is filled with a color and given a gray outline to enhance contrast between shapes of different sets. In addition, the shapes are dilated and eroded to smoothen the corners of the allocated element space and the transition between links and elements. A strong erosion results in a clear separation between shapes that are not nested (A2b), as seen at the top of Fig. 11.

Striped style. Striped Kelp uses alternating stripes for areas that contain elements of multiple sets (see bottom of Fig. 11). The allocated space $A(e)$ of $e \in \mathbb{E}$ is partitioned into radial slices, where every set that contains e gets the same number of slices in an alternating pattern. Edges of G_L that belong to a set have link radius r_l. When an edge belongs to multiple sets (links partially share a path), it is partitioned into stripes of fixed length where the sets give an alternating pattern.

Stripes are drawn as consistently as possible. If two links

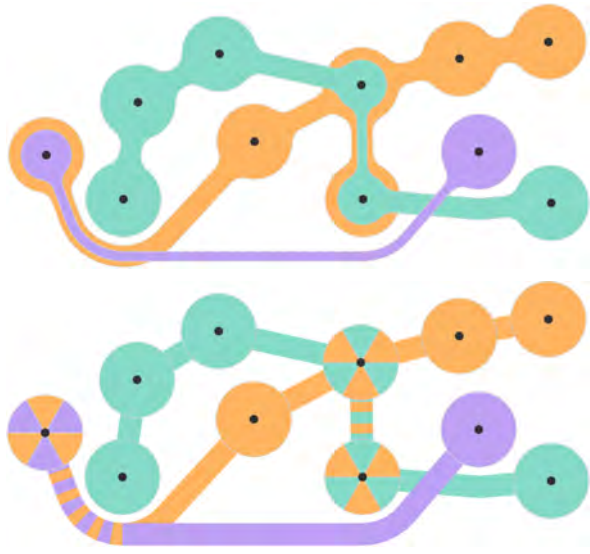


Figure 11: *Kelp Diagram of eleven elements and three sets. Top: Nested style. Bottom: Striped style.*

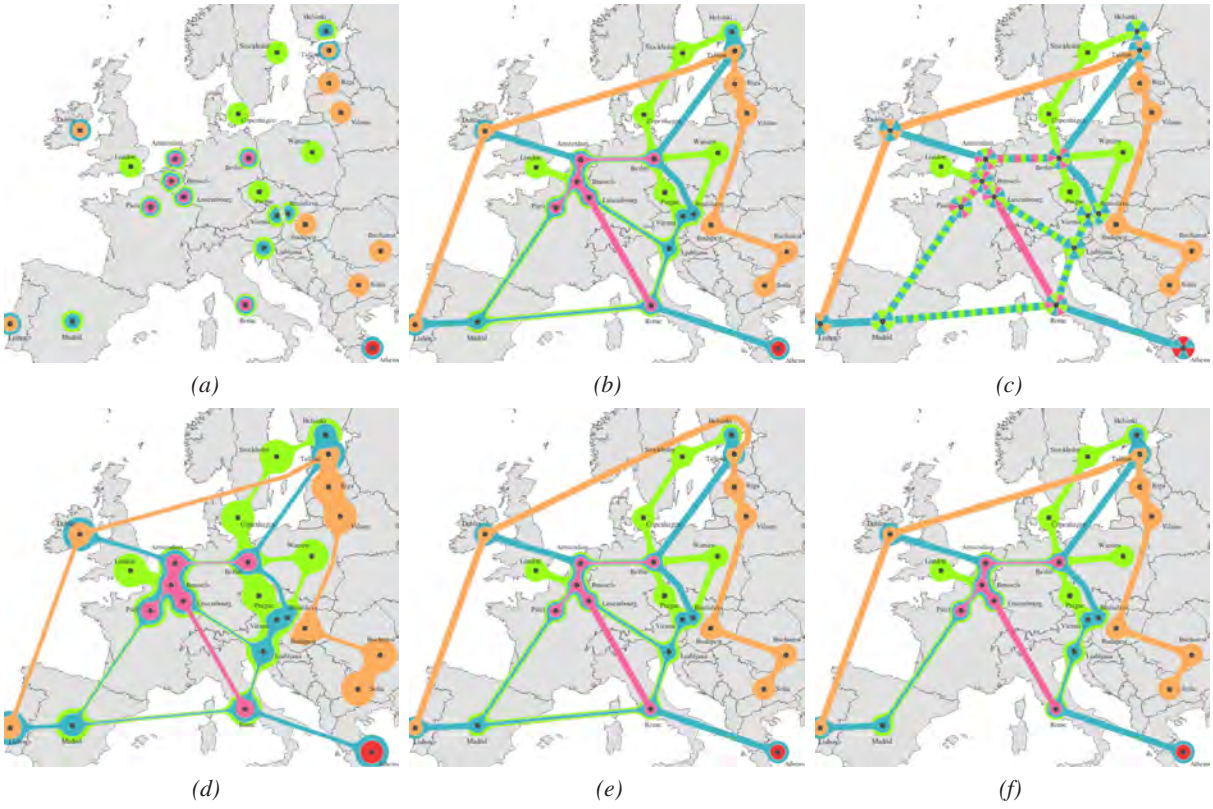


Figure 12: *Kelp applied to the capital cities of the European Union, where the eurozone is blue, the EU founding members (European Coal and Steel Community) are pink, and members with good, average, and bad credit rating are green, orange, and red respectively (Standard & Poor's, October 2011). The diagrams have various configurations: (a) Nested style without links; (b) Nested style with links; (c) Striped style; (d) Nested style with large element radius r_e and small link radius r_l; (e) Nested style with large intersection penalty c_I; (f) Nested style with low link addition threshold b_t.*

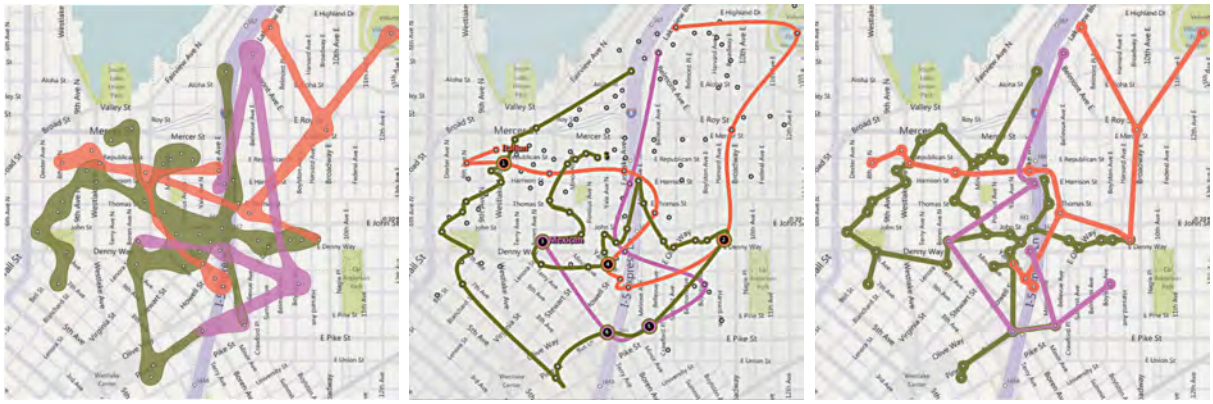
share edges but eventually split up, the stripes will continue along the edges at the split. This is achieved by drawing the stripes via a depth-first search in the G_L graphs. Certain cyclic configurations of links do not allow for consistent stripes, but these are rare enough in practice.

4.4. Implementation and performance

We implemented our approach in Java, using the Java Topology Suite [Viv03] for geometric operations such as dilation and erosion. One advantage of our purely geometric routing and diagram definitions is that vector graphics can be generated and merged with the predefined (vector) visualization to get images that can be rendered with arbitrary resolution.

The running time of the algorithm is dominated by the iterative placement of links. The tangent graph G_T contains $O(|\mathbb{E}|^2)$ edges and nodes. To place one link, shortest paths are computed to all nodes in G_T from every element and for every set. The same applies to all G_S graphs. This computation, including selection of the best link, amounts to $O(|\mathbb{S}||\mathbb{E}|^2 \log |\mathbb{E}|)$ when Dijkstra's algorithm is used. After a link is placed, any intersections that it introduces have to be accounted for. Thus, intersections are determined for all edge pairs, which amounts to $O(|\mathbb{E}|^2 \log |\mathbb{E}|)$ time with a sweepline algorithm. It is reasonable to assume that the algorithm's parameters are configured such that for every set $S \in \mathbb{S}$ the final $G_S(S)$ is planar and therefore $O(|\mathbb{E}|)$ links are placed for S. The entire links placement phase therefore takes $O(|\mathbb{S}|^2|\mathbb{E}|^3 \log |\mathbb{E}|)$ time. This also bounds the running time of the entire approach, with the space allocation and visualization phases being relatively cheap.

The bound (though not tight) indicates that the current approach cannot be applied to data sets of thousands of elements. However, most data sets commonly used in this visualization problem do not go beyond a hundred elements and several sets, since larger sets cannot be comprehended by an observer. Our implementation is able to generate Kelp diagrams for such data sets within five minutes on a modern desktop computer (Intel Core 2 Quad CPU at 2.4 GHz).



(a) (b) (c)

Figure 13: *Restaurants of three categories in Seattle: (a) Bubble Sets approach, generated with a public software library [KC]; (b) LineSets approach, image taken from [ARRC11]; (c) Nested Kelp Diagram.*

5. Results and Discussion

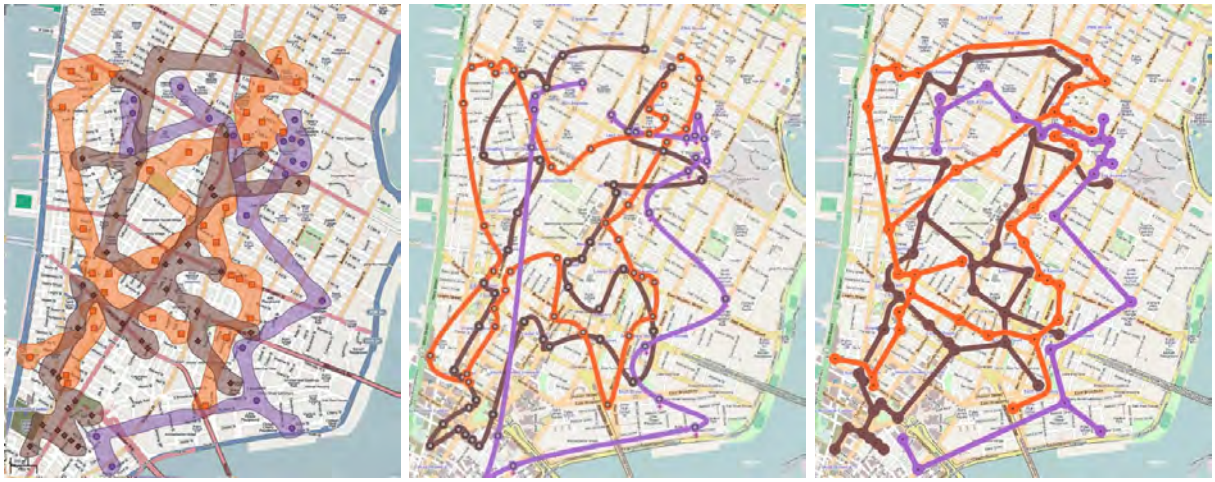
Fig. 12 shows Kelp Diagrams with various parameter configurations. The benefit of visually linking elements that belong to the same set follows from Fig. 12(a) and (b). Increasing the element radius r_e (see Fig. 12(b) and (d)) causes more of the original map to be occluded, making it harder to find capital cities by their location, but sets are more easily distinguished. Especially when multiple links share a path, the sets of the nested style are harder to distinguish, requiring larger r_l. The striped style suffers less from this. A comparison of Fig. 12(b) and (e) reveals the effect of increasing the intersection penalty c_I, where the algorithm deems it of greater benefit to route a long link around Helsinki instead of introducing additional intersections. The effect of the link addition threshold b_t is shown in Fig. 12(f), where a low b_t results in the construction of spanning trees. Kelp can be applied beyond cartography, as shown for a metabolic network in Fig. 1. Here, nested element scaling s_e has been given a very low value to push the contours away from labels of compounds.

The strengths and weaknesses of the nested and striped styles can be seen in Fig. 12(b) and (c). When multiple sets share the same link, e.g., Amsterdam-Berlin, the sets of the nested style quickly become harder to distinguish, whereas the striped style has no such problem provided that the link is long enough to fit ample stripes for each set. The striped style also prevents false assumptions to be made about the nested structure of the depicted sets. The nesting at Ljubljana suggests that the blue set is a subset of the green set, while they actually only overlap. However, the striped style ignores some of the criteria listed in Section 3. The stripes of a shared link or node break the continuation of the sets' shapes, which explains why the shapes of the nested style are more easily interpreted as a whole. Stripes also increase the visual complexity of the diagram because there are more and stronger bends (A1b). Therefore, the nested style is easier on the eyes and likely the more versatile style in many situations. It also happens to be the closest visual match to existing techniques.

Fig. 13 and 14 compare the nested style with the Bubble Sets and LineSets techniques for the same datasets. We can see that the Bubble Sets approach has many bends in its contours. In some locations, shapes and contours have undesirable overlap. Compared to the Kelp Diagram, the Bubble Sets depiction introduces more overlapping areas. This is not surprising since the Bubble Sets algorithm does not try to avoid intersections between links where Kelp does. In addition, Bubble Sets create shapes that use a lot of space and color blending introduces new colors for overlapping shapes that may be interpreted as additional, non-existing sets.

Bubble Sets and LineSets cannot visually connect elements beyond creating a spanning tree, whereas Kelp creates a graph to interconnect elements as much as desired via parameter b_t. Additional links, beyond a spanning tree, may not always be desired. For example, Fig. 14(c) has been generated with a relatively high b_t. For this diagram, some links of the brown set introduce some additional visual clutter (A1a, A1b, and A3a) though improve interpretation of spatial distributions (A2d and A3b). However, when b_t is set to a low value, surplus links will only be placed when continuation greatly improves. This is the case for the green set of Fig. 13.

LineSets do not route shapes around elements that should not be contained by them, which the other techniques do. Hence, in Fig. 14(b) we see an element of the brown set at the middle left that overlaps with a shape of the orange set, while it is not a part of the orange set. Moreover, LineSets connect elements of a set with a single line, creating longer links, many bends, and intersections. For example, there is a line with a strong bend at the bottom of Fig. 13(b) that could have been avoided.



(a) (b) (c)

Figure 14: *Disjoint sets of locations in Manhattan, with hotels (orange), subway stations (brown), and medical clinics (purple): (a) Bubble Sets, image taken from [CPC09]; (b) LineSets, image courtesy B. Alper and N. Riche; (c) Nested Kelp Diagram.*

Even though the element glyphs are very small in Fig. 13(c), we can immediately see the presence and density of elements because the surrounding contours are (partially) circular. For LineSets, the presence of an element can be inferred from the presence of a bend, while for Bubble Sets the large shapes make elements harder to spot (see Fig. 13(a)). Kelp's composition of basic visual units—circles and constant-width connections—is an advantage over Bubble Sets. Depictions are consistent, which means less visual clutter and easier interpretation.

Kelp Diagrams have some drawbacks. They sometimes have sharp bends where two links of the same set converge. Also, contours can become complex for clusters of elements that belong to the same set. LineSets suffer from this as well, in contrast to Bubble Sets. In addition, when elements are too close to each other, their Voronoi faces become dominant in space allocation and cause strong corners to occur in nearby contours. The schematic appearance of Kelp on top of a predefined visualization that is schematic as well can be confusing, e.g., the set shapes may be interpreted as roads or metro lines. The more organic appearance of LineSets and Bubble Sets make them stand out from a map.

6. Conclusion

Kelp Diagrams are a new way to depict set relations over already positioned elements. The algorithm balances visual complexity, based on aesthetic criteria, with effective depiction of the data. Comparison of resulting visualizations with those generated by two state of the art approaches for the same data shows that Kelp Diagrams have a consistent, easy to interpret, appearance. In addition, the parameters of the algorithm give it the flexibility that is required for application to different kinds of visualization.

Still, several improvements are possible. The routing algorithm is too slow for interactive use. Exploiting locality with spatial data structures, or replacing the tangent visibility graph with a graph of smaller complexity, are possible options. Also the greedy placement of links can be improved to reduce visual clutter. For nested Kelp, improved derivation of link width and nesting order with respect to aesthetic criteria requires further investigation. For example, links could be given widths according to an additive scheme, where the width of a link is increased when it has to convey multiple sets. In addition, allocated element and link space could be made dependent on the element density in an area.

Other extensions are possible, such as supporting elements with dimension (instead of points) and automated derivation of parameter settings based on the data itself. Use of the subset standard, as is the case for Kelp Diagrams, has been indicated to be effective via user studies before [ARRC11]. Nonetheless, we plan to investigate the effectiveness of the presented diagrams, and compare them to existing techniques, with further user studies.

Current approaches share a great deal of similarities in an attempt to solve the same problem. It would be interesting to explore a more generic method, which is able to produce the distinct visualizations of current approaches, but also able to produce hybrid visualizations of greater quality.

Acknowledgements. We would like to thank Nathalie Riche, Basak Alper, and their colleagues for providing us with the data for Fig. 13 and 14. B. Speckmann and K. Dinkla are supported by the Netherlands Organisation for Scientific Research (NWO) under projects no. 639.022.707 and no. 612.001.004, respectively.

References

[ARRC11] ALPER B., RICHE N., RAMOS G., CZERWINSKI M.: Design study of LineSets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2259–2267. doi:10.1109/TVCG.2011.186. 1, 3, 8, 9

[BCPS10a] BRANDES U., CORNELSEN S., PAMPEL B., SALLABERRY A.: Blocks of hypergraphs applied to hypergraphs and outerplanarity. In *21st Int. Workshop on Combinatorial Algorithms—IWOCA* (2010), Iliopoulos C. S., Smyth W. F., (Eds.), vol. 6460 of *Lecture Notes in Computer Science*, Springer, pp. 201–211. doi:10.1007/978-3-642-19222-7_21. 2

[BCPS10b] BRANDES U., CORNELSEN S., PAMPEL B., SALLABERRY A.: Path-based supports for hypergraphs. In *21st Int. Workshop on Combinatorial Algorithms—IWOCA* (2010), Iliopoulos C. S., Smyth W. F., (Eds.), vol. 6460 of *Lecture Notes in Computer Science*, Springer, pp. 20–33. doi:10.1016/j.jda.2011.12.009. 2

[BE01] BERTAULT F., EADES P.: Drawing hypergraphs in the subset standard. In *8th Int. Symp. on Graph Drawing* (2001), Marks J., (Ed.), vol. 1984 of *Lecture Notes in Computer Science*, Springer, pp. 45–76. doi:10.1007/3-540-44541-2_15. 2

[BT06] BYELAS H., TELEA A.: Visualization of areas of interest in software architecture diagrams. In *Proc. 2006 ACM Symp. on Software Visualization* (2006), Kraemer E., Burnett M. M., Diehl S., (Eds.), SOFTVIS, ACM, pp. 105–114. doi:10.1145/1148493.1148509. 2

[BT09] BYELAS H., TELEA A.: Towards realism in drawing areas of interest on architecture diagrams. *Journal of Visual Languages and Computing 20*, 2 (2009), 110–128. doi:10.1016/j.jvlc.2008.09.001. 2

[BvKM*10] BUCHIN K., VAN KREVELD M. J., MEIJER H., SPECKMANN B., VERBEEK K.: On planar supports for hypergraphs. In *17th Int. Symp. on Graph Drawing* (2010), Eppstein D., Gansner E. R., (Eds.), vol. 5849 of *Lecture Notes in Computer Science*, Springer, pp. 345–356. doi:10.1007/978-3-642-11805-0_33. 2

[CC07] COLLINS C., CARPENDALE S.: VisLink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1192–1199. doi:10.1109/TVCG.2007.70521. 2

[CPC09] COLLINS C., PENN G., CARPENDALE S.: 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. 1, 2, 3, 9

[dBCvKO08] DE BERG M., CHEONG O., VAN KREVELD M., OVERMARS M.: *Computational Geometry: Algorithms and Applications*. Springer, 2008. 6

[DBETT99] DI BATTISTA G., EADES P., TAMASSIA R., TOLLIS I.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999. 2, 3

[DTH08] DENT B., TORGUSON J., HODLER T.: *Cartography: Thematic Map Design*. McGraw-Hill, 2008. 6

[FMH08] FREILER W., MATKOVIC K., HAUSER H.: Interactive visual analysis of set-typed data. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1340–1347. doi:10.1109/TVCG.2008.144. 2

[FRM03] FLOWER J., RODGERS P., MUTTON P.: Layout metrics for Euler diagrams. In *Proc. Seventh Int. Conf. on Information Visualization* (2003), IEEE Computer Society, pp. 272–280. doi:10.1109/IV.2003.1217990. 2, 4

[HRD10] HENRY RICHE N., DWYER T.: Untangling Euler diagrams. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1090–1099. doi:10.1109/TVCG.2010.210. 2

[JP87] JOHNSON D. S., POLLAK H. O.: Hypergraph planarity and the complexity of drawing Venn diagrams. *Journal of Graph Theory 11*, 3 (1987), 309–325. doi:10.1002/jgt.3190110306. 2

[KC] KRAUSE J., COLLINS C.: Bubble sets library. URL: https://github.com/JosuaKrause/Bubble-Sets. 8

[KvKS09] KAUFMANN M., VAN KREVELD M. J., SPECKMANN B.: Subdivision drawings of hypergraphs. In *16th Int. Symp. on Graph Drawing* (2009), Tollis I. G., Patrignani M., (Eds.), vol. 5417 of *Lecture Notes in Computer Science*, Springer, pp. 396–407. doi:10.1007/978-3-642-00219-9_39. 2

[SA06] SHNEIDERMAN B., ARIS A.: Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 733–740. doi:10.1109/TVCG.2006.166. 2

[SAA09] SIMONETTO P., AUBER D., ARCHAMBAULT D.: Fully automatic visualisation of overlapping sets. *Computer Graphics Forum 28*, 3 (2009), 967–974. doi:10.1111/j.1467-8659.2009.01452.x. 2, 3

[SLK*09] STREIT M., LEX A., KALKUSCH M., ZATLOUKAL K., SCHMALSTIEG D.: Caleydo: connecting pathways and gene expression. *Bioinformatics 25*, 20 (2009), 2760–2761. doi:10.1093/bioinformatics/btp432. 2

[ST10] SANTAMARÍA R., THERÓN R.: Visualization of intersecting groups based on hypergraphs. *IEICE Transactions on Information and Systems 93-D*, 7 (2010), 1957–1964. 2

[SWS*11] STEINBERGER M., WALDNER M., STREIT M., LEX A., SCHMALSTIEG D.: Context-preserving visual links. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2249–2258. doi:10.1109/TVCG.2011.183. 2, 3

[vDvKSW02] VAN DIJK S., VAN KREVELD M. J., STRIJK T., WOLFF A.: Towards an evaluation of quality for names placement methods. *International Journal of Geographical Information Science 16*, 7 (2002), 641–661. doi:10.1080/13658810210138742. 4

[Viv03] VIVID SOLUTIONS: Java topology suite, 2003. 7

[WvdBH07] WEIN R., VAN DEN BERG J. P., HALPERIN D.: The visibility-Voronoi complex and its applications. *Computational Geometry 36*, 1 (2007), 66–87. doi:10.1016/j.comgeo.2005.11.007. 5