Bubble Sets: Revealing Set Relations with Isocontours over Existing Visualizations

Christopher Collins, Gerald Penn, and Sheelagh Carpendale

Abstract—While many data sets contain multiple relationships, depicting more than one data relationship within a single visualization is challenging. We introduce Bubble Sets as a visualization technique for data that has both a primary data relation with a semantically significant spatial organization and a significant set membership relation in which members of the same set are not necessarily adjacent in the primary layout. In order to maintain the spatial rights of the primary data relation, we avoid layout adjustment techniques that improve set cluster continuity and density. Instead, we use a continuous, possibly concave, isocontour to delineate set membership, without disrupting the primary layout. Optimizations minimize cluster overlap and provide for calculation of the isocontours at interactive speeds. Case studies show how this technique can be used to indicate multiple sets on a variety of common visualizations.

Index Terms—clustering, spatial layout, graph visualization, tree visualization.

1 Introduction

Many types of data that are important to analysts, such as social network data, geographical data, text data, and statistical data, often contain multiple types of relationships. These can include *set* relations that group many data items into a category, *connection* relations amongst pairs of data items, *ordered* relations, *quantitative* relations, and *spatially explicit* relations such as positions on geographic maps. Common approaches to visualizing set data focus on solutions that integrate clustering and bounding outlines. That is, when possible, set members are moved into close proximity and contained within a convex hull. When set member proximity is not possible, alternate approaches make use of additional visual attributes, such as colour, symbol or texture, to indicate that discontinuous items or groups are in fact members of the same set. Bubble Sets provide continuous bounding contours, creating outlines analogous to hand-drawn enclosures, and do not require prior spatial reorganizing. This eliminates the cognitive load of keeping track of spatially discontinuous set members.

Bubble Sets is an approach to visualizing set relations over existing visualizations that respects the spatial rights of the initial visualization. For some types of data, such as scatter plots, maps, and timelines, the semantics of the spatial variable is crucial. For other types of data, such as node-link graphs, one could argue that spatially arrangements can be re-organized, and while we agree that this can be the case there are also times when a particular layout is beneficial for a task, and spatial re-organization can interfere with one's mental model [16]. Therefore, in Bubble Sets, we respect the spatial rights of the primary visualized relation and do not disturb it when displaying the secondary set relations. Our approach uses implicit surfaces to create continuous and dynamic hulls which highlight set relations without requiring layout adjustments to existing visualizations. Heuristic optimizations allow for set boundaries to be calculated at interactive speeds and to guarantee set continuity while minimizing overlap. The Bubble Set approach to set visualization can readily be applied over any existing visualization, be it spatially explicit or spatially assigned.

In the following section we advocate for careful consideration of spatial rights when visualizing multiple relations and review the related work using spatial rights as a means to differentiate between the types of visualizations that contain both connection and set relations. Then we outline our algorithmic approach to determining and rendering set boundaries, and illustrate our technique with four case studies.

- *Christopher Collins and Gerald Penn are with the University of Toronto, E-mail: {ccollins,gpenn}@cs.utoronto.ca.*
- *Sheelagh Carpendale is with the University of Calgary, E-mail: sheelagh@cpsc.ucalgary.ca*

Manuscript received 31 March 2009; accepted 27 July 2009; posted online 11 October 2009; mailed on 5 October 2009.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org .

2 Background

We use the common *set* definition: a collection of unordered items, possibly empty, with no repeated values. This includes Freiler *et al.*'s definition of set membership by *set-type attribute* [11]. Also, we include sets that are defined implicitly by relationships amongst members (a group of friends), sets which share data attribute values (all cars with air conditioning), and sets which are arbitrarily specified (personally-selected).

One approach to set visualization is to consider the set relation as primary and create a spatial layout according to set membership [11]. In this situation spatial proximity within sets can be achieved. Another approach is to spatially adjust or re-cluster a given visualization to bring set members into closer proximity, making it possible to visually group them with a convex hull. Since both of the above approaches result in spatially clustered set members, visually containing them in convex hulls can be effective [8, 13, 17]. Convex hulls are fast to calculate and well-suited to cohesive clusters which are separated from neighbouring groups. However, if the layout contains items that are not set members but are within the set region spatially, these items appear within the convex region determined by set members, thus will appear to be set members. This is seen in ScatterDice [9] when scatter plot axes change after sets are defined with the lasso tool.

Additionally, in some spatially assigned layouts, such as scatter plots, and in other data representations, such as maps, the semantics of the layout preclude spatial re-positioning. In these situations use of convex hulls to encircle set members is not effective. Here set membership is sometimes indicated through discontinuous set outlines and/or the use of colour and symbols. To provide a method that can visually contain set members within an outline, we calculate a polymorph hull which can have convex and concave regions. This hull can avoid including items which are not set members, except in very dense arrangements where a human would also have difficulty manually drawing the enclosure. We use implicit surfaces similar to those used by Watanabe *et al.* [22] and apply them to create contiguous multi-set visualizations over multi-relational information visualizations. The requirement for more complex hull outlines to support set visualization within many different types of visual representations is closely tied to the spatial semantics and the concept of spatial rights.

2.1 Spatial Rights

Since research into how we perceive visualizations indicates that the spatial positioning of data items may be the most salient of the possible visual encodings such as position, colour, shape, *etc.* [3, 21], it may well be important to preserve positioning of data items while providing

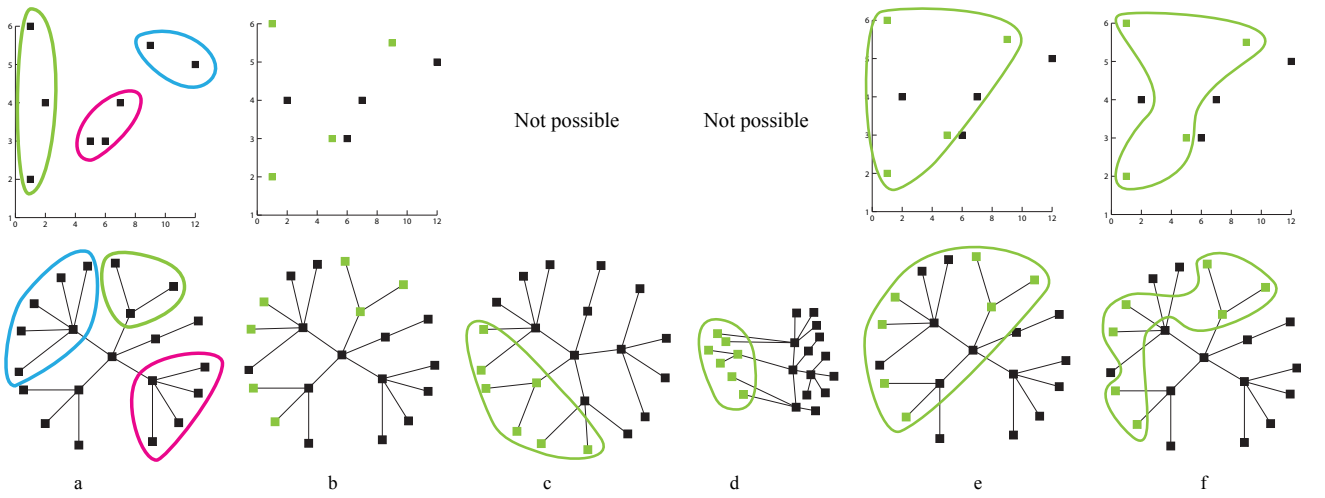

Fig. 1: Rows — Set relations over statistical plots (top row) and node-link graphs (bottom row). Columns — (a) Simultaneous spatial rights: set membership is based on proximity in layout. (b) Spatial rights for the connection relation: set members indicated through common item colour. (c) Hybrid spatial rights: layout based on connection relation is adjusted to bring set members into closer proximity. (d) Spatial rights for set relation: layout brings set members into a tight cluster, obscuring the structure of the connection relation. (e–f) Spatial rights for attribute-based layouts and connection relations: the set relation is drawn atop existing layouts using either the traditional convex hull (e) or Bubble Sets (implicit surfaces) (f).

set membership visual containers. We refer to this concept of granting primacy to the particular aspect of the data on which the layout is based as *spatial rights* [7].

Social network data exemplify the different types of relations we consider. These sorts of rich data sources are important to understanding organizations, online culture, and computer-based collaboration. A connection relation may be that friends are directly connected to one another, or that supervisors are connected to their direct reports in a management hierarchy. A set relation could be defined in several ways: based on graph characteristics such as cliques of closely connected friends, based on demographic information such as occupation or level of education, or arbitrarily determined, for example, by an analyst. *Set* relations are sometimes called *categorical* relations, where all items sharing a value for a categorical data attribute comprise a set. Finally, additional ordered and quantitative attributes may be present, and could be used to annotate or position the visual items, such as arranging by date of birth or annual income. With these data, it is likely most intuitive to provide a layout based on the connectedness information, as that is the 'network' in social network analysis. In this case, it may not be possible or desirable to adjust the layout to maximize set contiguity or density. Doing so may disrupt the meaning provided by the connection-based layout. Traditional set visualization techniques will not be adequate, and our approach may be more applicable.

Traditionally, either the connection relation or the set relation is afforded primacy, and the spatial layout of a visualization is based on that relation. There is a trade-off in design — optimized layouts for connections may lead to unclear sets; optimized layouts for sets may lead to confusing edge crossings and a loss of visible structure. An alternative is to design a coordinated view system, in which one view is based on the set relation, and the other is based on the connection relation. However, in this work we will restrict ourselves to tightly coupled relations in a single-view visualization. Our implicit surface solution clarifies set membership while not disrupting information-carrying layouts based on connection relations and ordered attributes. We will explore the related work by defining five configurations of spatial rights and giving examples from the literature where possible. Figure 1 illustrates this discussion by showing in diagrams different ways to indicate set membership in scatterplots and on tree layouts.

Simultaneous Spatial Rights: If a dataset contains one set relation and one connection relation, it may be spatialized based on either. One may desire contiguous, compact clusters to represent sets, or one may wish to design a layout to minimize overlap of edges in the connection relation. These two goals are not always mutually exclusive. In some cases, set membership is a direct function of the connectedness characteristics of a graph, or even the spatial proximity resulting from a layout based on graph structure. In these cases the desired layout can be created based on the connection relation, and sets will appear closely clustered and contiguous, giving both spatial rights. Figure Figure 1a shows a scatter plot (top) and a tree (bottom) where each set's spatial proximity allows for bounding by a convex hull. Examples of this include the friend clusters in Vizster [13] or spatially-determined aggregates in level-of-detail visualizations [1]. Our investigation of the design space of multi-relationship visualizations reveals that assigning simultaneous spatial rights is not always possible.

Hybrid Spatial Rights: The set relation may be pre-determined by data characteristics or set membership may be arbitrarily selected by an analyst. In these cases the layout may be adjusted to bring set members closer together. In this condition, the connection relation and set relation share spatial rights in a hybrid layout. However, reorganizing a layout to create more continuous and denser sets may have serious consequences for the readability of the primary connection relation. For example, nodes in a tree may have a meaningful order which reorganization may disrupt. In Figure 1b for both the tree and scatterplots set membership is discontinuous and is indicated by node colouring. In Figure 1c the tree from Figure 1b is re-organized to provide set proximity, however, this re-organization would destroy meaning in the scatterplot. While we do not know of examples which explicitly create a hybrid layout for set and connection relations, Phan *et al.* [18] report a hybrid layout based on hierarchical clustering and adjusted to provide node separation. Dwyer *et al.* [8] use a hybrid layout technique to provide for fast calculation of large-scale overviews which are adjusted in detailed views for higher-quality layout.

Spatial Rights for Set Relations: When the set relation is the primary relation, it can be assigned spatial rights. Items are then arranged to maximize set separation, continuity, and density. Connection relations are drawn atop this layout (Figure 1d), however, this approach does not apply to scatterplots. We do not know of any research that affords primary spatial rights to a set relation while drawing a connection relation atop. The closest analogues are approaches using multiple connection relations, such as using hierarchical data to lay a treemap out, then drawing additional connection relations atop [10].

Spatial Rights for Connection Relation: When the connection relation is the primary relation, it is best to assign it spatial rights. When set membership is not based on the connection relation, but rather on an unrelated data attribute, or even interactive selection, common visualization techniques such as convex hulls are not sufficient (Figure 1e).

Visualizing set relations for data items in a predetermined visual layout is often difficult due to spatial discontinuities, and set overlaps. Set-membership ambiguities can be introduced by the use of convex hull algorithms. Noting set membership with another visual encoding such as symbols or colours on the data items is helpful but lacks the clarity of a single enclosure. Indicating set membership with 'bubbles' — contours that tightly wrap set members — has been attempted in the past, however, this approach only offered continuous set membership enclosure for proximal groupings [14, 22]. Byelas and Telea [6] provide connected set enclosure visualizations for UML using outer skeleton construction and handle incorrectly included items by second pass cutouts. In this work, we introduce an efficient implementation of implicit surfaces which allows for contiguous sets to be drawn over arbitrary layouts, while reducing set membership ambiguity problems using interactive highlighting (Figure 1f).

Data explicit Spatial Rights: With some data types such as maps the spatial rights are explicit in the data semantics. Here set memberships can exist across multiple distances such as the sets of all capitol cities or the cities with populations over a million. With this type of data spatial re-organizing is not an option and solutions such our approach diagrammed in Figure 1f are essential.

2.2 Implicit Surfaces

While implicit surfaces have been previously used to illustrate set relations over graphical objects [14, 22], in both works, the set relation is defined by the spatial proximity of graphical objects — increasing an item's distance from the set centroid removes the item from the set; pushing an item toward the set centroid adds it. As we aim to maintain the visualization of set membership for any spatial relationship between set members, dragging an item away from a set centroid will not remove it from the set. Thus we provide alternative interaction techniques to modify set membership. Watanabe *et al.* [22] provide no option for set members existing across long distances. Heine and Scheuermann [14] allow for a single set to be divided into multiple subgroups separated by distances by using bubbles of the same colour. They do not detect an isocontour, rather using pixel-based shading to display sets. Thus rich interaction with the set is not possible. In our work, we maintain a continuous and connected contour around all set members irrespective of distance or spatial organization.

3 Interactive Bubble Sets

Our approach arose from observing curved and complex boundaries hand-drawn by people to indicate set relations (Section 4). To simulate these natural-looking boundaries, our method requires: (a) all set members to be enclosed, (b) non-members to be excluded, (c) where non-members occur within boundaries, visual and interactive hints to clarify membership, (d) rendering to allow for interactive adjustment.

Implicit surfaces, more accurately called 'contours' in 2D, are well suited to address these requirements [4]. In this section we will describe our version of implicit contours ('bubbles') and the heuristics we employ to fulfill the requirements and create accurate and aesthetically pleasing bubbles around set members. An implicit surface is simply a contour such that the energy $E(x,y) = c$ where c is a constant potential value. We consider the display space to be a grid upon which we calculate potential 'energy' values for each cell (pixel). When the potential energy function is continuous, continuous contours are guaranteed. However, there may be more than one separate contour in the potential grid. Contours are defined by the presence of items on the grid. An item is an object that may or may not be a set member. For each pixel the potential energy is the sum of influences of nearby items, as a function of distance:

$$s_{pixel} = \{j | j \in items, distance_{j,pixel} < R_1\} \quad (1)$$

$$energy(pixel) = \sum_{i \in s_{pixel}} w_i (R_1 - distance_{i,pixel})^2 / (R_1 - R_0)^2 \quad (2)$$

where R_0 is the distance at which energy is 1, R_1 is the distance at which energy reaches 0, w is the weight assigned to the item, *items* is the set of all items in the space, and s_{pixel} is the set of influencing items within R_1 of the pixel. An isolated point item, then, will have a circular isopotential contour at radius R_0 and an energy field extent of R_1. As items are often not points, but rather shapes such as rectangles, we use Euclidean distance to the nearest point on the shape surface. Inside shapes we assign $distance = 0$. As the energy function reaches its root at R_1, only items within R_1 of a pixel are included in the energy calculation and will have a non-zero effect. After calculating energy values for all grid cells, we use a 2D version of Marching Cubes [15] to trace the contour. As described in the following sections, we adapted this general method with additional steps to ensure sets are connected and contain all set members, while excluding items not in the set. The simplified algorithm for calculating bubble boundaries is described in Algorithm 1.

Algorithm 1 Determining a Bubble Set Boundary

```
given items with positions
for all sets s ∈ S do
    find centroid c of s
    for all items i ∈ s, order ascending by distance to c do
        find optimal neighbour j ∈ s
        find best route from i to j
        for all cells (pixel or pixel group) within R1 of i do
            add potential due to i
            add potential due to nearest virtual edge i → j
            subtract potential due to nearby non-set members k ∉ s
        end for
    end for
    repeat
        perform marching squares to discover isopotential contour s̄
        reduce threshold
    until ∀i ∈ s, isocontour s̄ contains(i)
    draw cardinal splines using every Nth point on the contour
end for
```

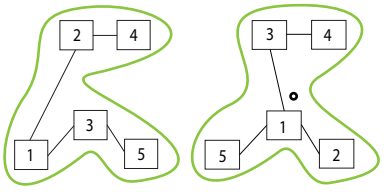

Fig. 2: The order of connecting set members with virtual edges affects the generated shape. Left-to-right, top-to-bottom connection generates a snake-like virtual edge configuration (left), while connecting from the centroid (black circle) outward generates a more blobby shape (right).

3.1 Surface Routing

We generate *virtual edges*, which are routed around obstacles to allow set surfaces to 'flow' in an aesthetically pleasing way while avoiding overlaps with nodes and maintaining set connectivity. For each set, we first define an *active region* as the rectangular bounding box which includes all set members, increased on all sides by a buffer of R_1. Only items in the active region can be close enough to set members to affect the bubble creation. Therefore, for speed purposes, items and pixels outside the active region are not included in the energy calculations.

There are two options for routing the bubble surface. First, if *structural edges* (edges that are part of the data) are included, the bubble surface should follow them. For example, if a set over a node-link graph includes both nodes and edges, we simply use these items to calculate the bubble surface. However, as discussed previously, set relations may not have any dependence on a connection relation in the data. In these cases, we ignore any edge structure in the visualization and determine bubble routing based on the constraint that bubbles must, where possible, avoid overlapping or including non-set members within the bubble boundary. To achieve this, we route the bubble surface around items which should not be enclosed using an invisible backbone of edges that connect set members while avoiding

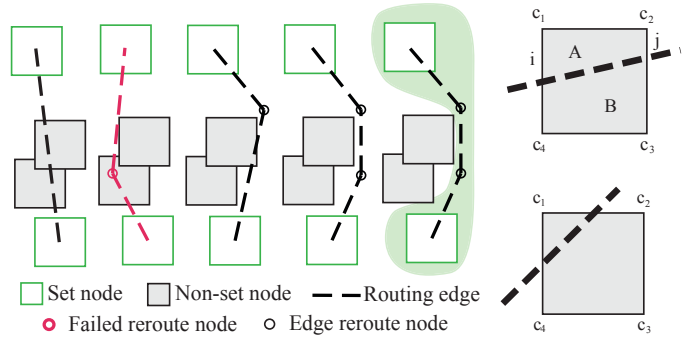

Fig. 3: Contour connectedness is assured through virtual edges which add to the energy distribution for the set. (l-r): A virtual edge passing through a node is detected. A new control point is created at a corner of the obstacle's bounding box and the test repeated. As the test fails, the diagonally opposite corner is then used and no obstacle is found. Additional control points are created at the corners to route edge around the obstacle. The final set of virtual edges contributes to the energy calculation, allowing the set contour to avoid the obstacles and remain connected. Far right: the configurations for creating virtual nodes with Algorithm 2.

non-included items. This backbone is initialized by iterating through set members and connecting them to an optimal already visited member of the same set using a straight line between item centers. The optimal neighbour $j_{optimal}$ for node i is selected to minimize the function $cost(j) = distance(i,j) * obstacles(i,j)$ where $obstacles(i,j)$ is the count of non-set members on the direct path between i and j. This function balances a preference for close connections with the simplicity of straighter paths. We start this operation with the item nearest the set centroid and proceed outward. This encourages 'blob' shapes rather than 'snake' shapes (Figure 2). This process ensures all set members are connected.

Extending the edge routing algorithm of flow maps [18], we then test all non-set members within the active region for intersection with the virtual edges. If an intersection is found, we split the edge and route it around the blocking node by creating a new control point offset from one of the node corners and connecting the original edge endpoints to this control point, creating 2 virtual edges. We place the control point R_1 away from the corner to provide for a buffer around the blocking item. If the creation of the new control point initially fails, we iteratively try routing around other corners of the obstacle and reducing the buffer. This process, detailed in Figure 3 and Algorithm 2, is repeated for the new segments until an iteration limit is reached or no intersections are found. This method does not work for the case where a set member's bounds are completely contained within a non-set member (see scatter plot case study). Our algorithm is $O(KN(N+HW))$ for K sets, N items, and a pixel field of H by W, however heuristics such as restricting energy calculations to the active region around an item reduce the average case considerably.

3.2 Energy Calculation

To calculate the potential field, several techniques are employed to gain speed. First, the display space is divided into square pixel groups which are treated as a single pixel. While this lowers the resolution of the surface calculation, the visual artifacts introduced are minimal, and it actually has the effect of smoothing the bubble surface. We dynamically adjust the pixel group size to provide interactivity when items are dragged (pixel group 9×9) and higher quality rendering when the scene is static (pixel group 3×3).

The potential field is calculated for each set in sequence. Nodes and edges (both structural and virtual) in the set are given a positive weight of 1. Negative energy influences result in the implicit surface being pushed away from items not included in the set. Nodes not included in the set are weighted -0.8. Non-set edges are usually given a weight of 0, as bubbles will not be able to avoid edge crossings, and energy reductions at edge crossings can cause surface discontinuities. The

Algorithm 2 Route virtual edges around obstacles (see Figure 3).

```
while ∃ virtual edge l_{k,m} which intersects obstacle do
  n ← null, swap ← false
  while (n = null ∨ n intersects obstacle) ∧ (buffer > 0) do
    if l_{k,m} intersects adjacent edges of obstacle bounds then
      add virtual node n at corner of adjacent edges
    else
      if Area(A) ≤ Area(B) then
        if i > j then
          add virtual node n at (swap ? c1 : c3) + buffer
        else
          add virtual node n at (swap ? c2 : c4) + buffer
        end if
      else
        if i > j then
          add virtual node n at (swap ? c3 : c1) + buffer
        else
          add virtual node n at (swap ? c4 : c2) + buffer
        end if
      end if
    end if
    if swap then
      reduce buffer
    end if
    swap ← ¬swap
  end while
  split virtual edge into l_{k,n} and l_{n,m}
  reset buffer
end while
```

energy contribution of visual items is also dependent on R_0, and R_1 — these parameters must be tuned for a particular application depending on the size and spacing of items, and the bubble margins desired.

For a given set, we first calculate the positive energy influences for each pixel group in the active region. That is, for all set members and virtual edge sets, we calculate energy contributions for all pixel groups within R_1 of the item. For a given pixel group and virtual edge set (route between two nodes), only the edge segment closest to the pixel group contributes energy. This avoids overly large positive energy values near segment connection points, which would lead to bulges in the final surface. Next, for pixel groups with total energy greater than zero, we add the negative influence of entities which are not in the set. As regions of zero or negative energy will never be part of the isocontour, we do not calculate negative energy contributions unless the pixel group already has a positive energy. This provides a significant reduction in the time required to fill the energy surface of the active region. A visualization of the energy field underlying a set is shown in Figure 4.

3.3 Contour Discovery

We used a 2D version of Marching Cubes [15] to discover an isocontour for each set. After discovering the isocontour, we check all set members to ensure their centers are within the contour enclosure. If they are not, this indicates a disconnection. In this case, we iteratively reduce the potential threshold by a factor *alpha*, repeating marching squares until all members are included. In very dense layouts, it may be necessary for the set contour to pass through items which are not included in the set, for example if items are adjacent without space for routing around. In practice, such situations would be difficult for a person to draw — if no space is available for the set to route around an obstacle, our algorithm will eventually go through it. The marching squares step is fast to repeat $O(H+W)$ for an active region of H by W pixel groups. If, however, after N iterations the set is still disconnected, we manipulate the energy field, a slower process ($O(HWK)$ for K items in the active region). We increase the positive weights by a factor β, decrease the negative weights by a factor γ and recalculate the potential field, followed by another iteration of marching squares. We repeat the energy manipulation for N additional iterations or until the set boundary encloses all member items.

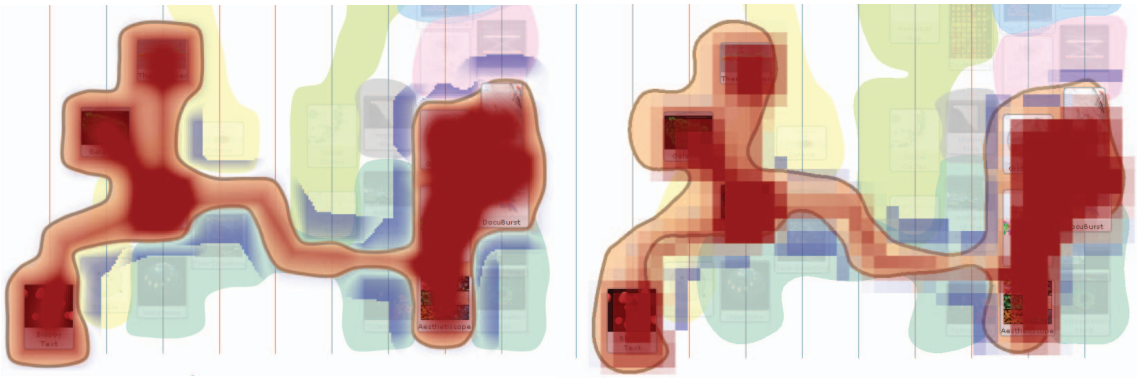

Fig. 4: Visualization of potential energy field for a single set. At left, with pixel group = 1 for high quality rendering, at right with pixel group = 9 for interactive animation. Non-member items and additional sets are faded in the background. Red areas indicate positive potential, blue areas are negative potential. Note the visible bends in the virtual edges route in the center and far left, an effect of the routing algorithm. The isocontour is shown as a solid brown boundary.

Negative energy contributions serve to 'push' the set boundary away from non-set members near the contour boundary. Surface routing attempts to work the boundary around obstacles, but when a dense set encloses a non-set member, the algorithm may fail to exclude that item. If a non-set member is discovered within the contour, we highlight it with a white border. It would also be possible to create a hole using an iteration of marching squares beginning at the center of the non-set member and working outward to discover an interior contour. We leave this for future work and use highlighting and interactivity to clarify such occurrences.

To render the surface we use cardinal splines, using every Lth point on the surface discovered by marching squares as a control point. The selection of L involves a trade-off between smoothing and precision, and is dependent also on the pixel group size. We lean toward smoother surfaces and select $L = 10$ in the examples that follow. We colour the interiors of the surfaces with a transparent version of the border colour to more clearly indicate the extent of the set region.

3.4 Interaction

Interactions such as adding items to sets, removing items from sets, creating new sets and moving items within sets are provided. Since proximity does not determine set membership, we use moded interaction. First a set is made active with a right-click, then individual items can be added or removed by clicking them. When complete, the set is deactivated and its new contents are fixed. Depending on the application requirements, it is also possible to select and move an entire set by clicking on the set background. Our requirement for surface calculation and rendering at interactive speeds is met, as discussed, by defining active regions, pixel groupings, and only calculating negative influence for pixels with energy greater than zero. Sets are rendered from largest to smallest to facilitate picking, ensuring smaller sets are not completely covered. After the initial rendering, we only recalculate the contours for sets with changing configurations. Specifically, for any moved item i, we only recalculate the surfaces for sets $s \in S$ if i is in the active region of s. The speed gain of this heuristic is dependent on the density of the display space and the number of sets.

We cannot guarantee that non-set member items will be excluded from the set boundary in all cases. The use of surface routing and negative energy influences minimizes this occurrence, but it remains possible depending on the density of the graph and the particular layout of items. To clarify set membership, we visually separate overlapping or enclosed non-set items from the set using a white border (*e. g.*, the gold item overlapping the green set near the center of Figure 9). Additionally, when a set member or set boundary is under the mouse, all non-set member items are faded out (made partially transparent).

4 Case Studies

To demonstrate the flexibility of our technique, we have applied it to visualize set relations over several visualizations for which spatial rights are given to the existing layout that contains the set members. Our examples include maintaining spatial rights for the connection relation in the machine translation parse trees, hybrid spatial rights in the timeline visualization, explicit spatial rights in the map example, and ordered spatial rights in the scatterplot. Our isocontour surface calculation and rendering was implemented in Java as an extension to the prefuse toolkit [12]. The case studies are prefuse-based visualizations using isocontour surfaces to display set relations.

4.1 Machine Translation Parse Trees

As a first example, we have developed a visualization for statistical machine translation data. The task of our translation researcher collaborators is to examine the outputs of the translation system in order to discover any problems in the training data that lead to translation errors. Their technique generates translations by translating segments of a source sentence into fragments of a linguistic parse tree in the target language. The collection of possible parse fragments is then assembled to create a complete parse tree. The translated sentence is read from the leaves through in-order traversal.

This statistical syntax-based machine translation system operates with two probability models: a translation model and a language model. The translation model assigns probabilities that a given sentence segment will translate to a particular parse tree fragment. The language model assigns likelihoods that the candidate translation is a valid sentence in the target language. The two models combine to result in an overall score assigned to a translation candidate. The task of the researchers is to diagnose translation problems that are most often the result of invalid translation pairs in the translation model's training data. To do this, analysts examine hundreds of translation examples, manually tracing problems in the translation back to the source segment and target parse tree fragment pair that caused the problem.

The observed computational linguistic researchers' diagnostic process consists of two steps: first, review an entire (printed) parse tree for problems, second, if problems are found, scan through several pages of tree fragment-translation pairs to discover problem sources. In their process, there was no way to see which nodes in the visualized parse tree consisted of a set or fragment. However, when sketching on their print-outs, analysts often drew bubbles around problematic tree fragments in the parse. The data for the visualization consisted of a connection relation (parse tree) and a set relation indicating which tree fragments were a unit of translation. We assigned spatial rights to the connection relation, laying out the parse tree with an improved version of Reingold and Tilford's layout [5]. The set relations are not determined by proximity in the parse tree, but rather specified by the way in which the tree is constructed from fragments. However, the set relation is not completely independent of tree structure — individual sets are guaranteed to be connected through tree edges. Thus, the structural edges rather than our virtual edges are used for surface routing.

Our set visualization approach is well-suited to reveal the tree fragments directly on the output parse tree. Set background hue is selected

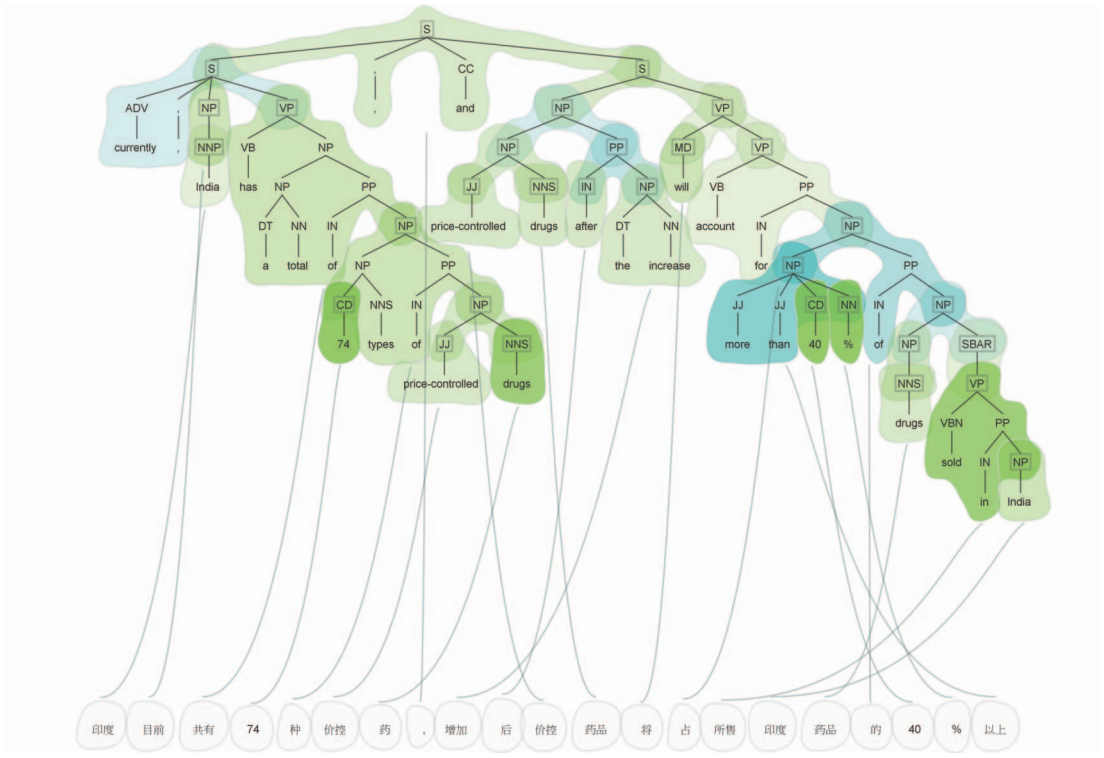

Fig. 5: An example translation parse tree from Chinese to English. Bubble colour indicates the type of translation rule, increasing bubble opacity indicates a higher probability score. Edges indicate correspondence between sets of the English parse tree and Chinese words.

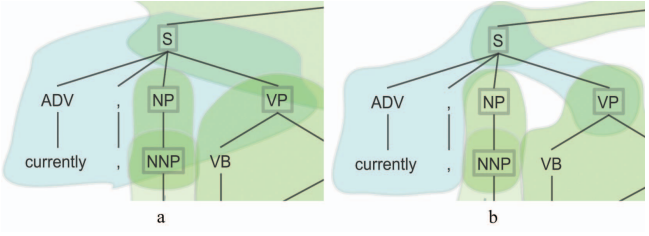

Fig. 6: Different surface drawing techniques: (a) Blue convex hull includes non-set members within the boundary, (b) Bubble Set does not.

based on the category of the fragment (a classification important to the analysts), and the background transparency is based on the confidence score assigned to the set by the translation algorithm (darker is more confident) (Figure 5). Traditional methods such as convex hulls would not suit this application, as they are unable to exclude non-set members (Figure 6). As part of a tool for analyzing translation models, our visualization allows translation researchers to review translation parse trees for problems and annotate discoveries directly on the visualization, without the need for lengthy tables of translation tree fragments. We are planning to deploy the system for a longitudinal case study.

4.2 Research Articles Timeline

Spatial tools to organize personal archives of PDF documents have been reported, including the Dynapad tool which provides both clumped (set-based) and timeline (attribute-value-based) layouts [2]. However, Dynapad is not able to display the set relation when the PDF article icons are displayed in the timeline formation. Combined timeline and set views of the research literature are used for personal information organization, and for communicating an organization applied to a set of articles [20, p. 23]. Several people were involved with the discussion about which articles belonged in each set, and the groupings were revised several times. Inspired by this activity and the manually-created sketches [20], we provide an automatic creation of similar visualizations as a tool to support collaborative discussion and categorization of research articles (Figure 7).

In our prototype implementation, we use the data from [20]: a collection of 60 research articles in the area of visualizing communication patterns. We manually collected characteristic images and abstracts from the electronic documents, but document thumbnails could easily be generated using the PDF icon generation method of [2]. The layout used in this example provides hybrid spatial rights: the initial layout places article icons on a timeline according to the year of publication. The layout is then adjusted to improve density of the predetermined set memberships using a force-directed layout algorithm, restricted to movement on the y-axis. Forces draw items that share at least one set toward one another and repel members of other sets. This layout mechanism provides for dense sets with minimal interference from other sets. However, in years where many articles appear on the timeline, interference cannot be avoided. In these cases, the surface routing algorithm creates an invisible structure of edges to route the set surfaces around obstacles.

Interactive movement of items along the y-axis allows for creation of customized views. Set membership is specified in the dataset, but can be changed at run time. To aid interactive classification of articles, the data can be queried for additional details, such as a larger image or the abstract, by clicking (desktop) or tapping (interactive tabletop) a document icon. The set boundary expands to contain the enlarged item, and a force-directed algorithm moves neighbouring items to maintain partial visibility (Figure 8).

4.3 Sets over Geographical Maps

Grouping items over spatially explicit data, such as geographical maps, is especially challenging for a convex hull approach. Maps commonly have widely dispersed sets, such as cities, rail stations, or lakes. In cartography, these categorical sets are usually identified by coloured or iconographic symbols. However, due to the distance between symbols, the visual interference of other map features, and their relatively small size, it is unlikely that sets of similar symbols are perceptually associated quickly. Consider a scenario in which a health-care jobseeker is selecting a hotel close to transit and interview sites. Bubble

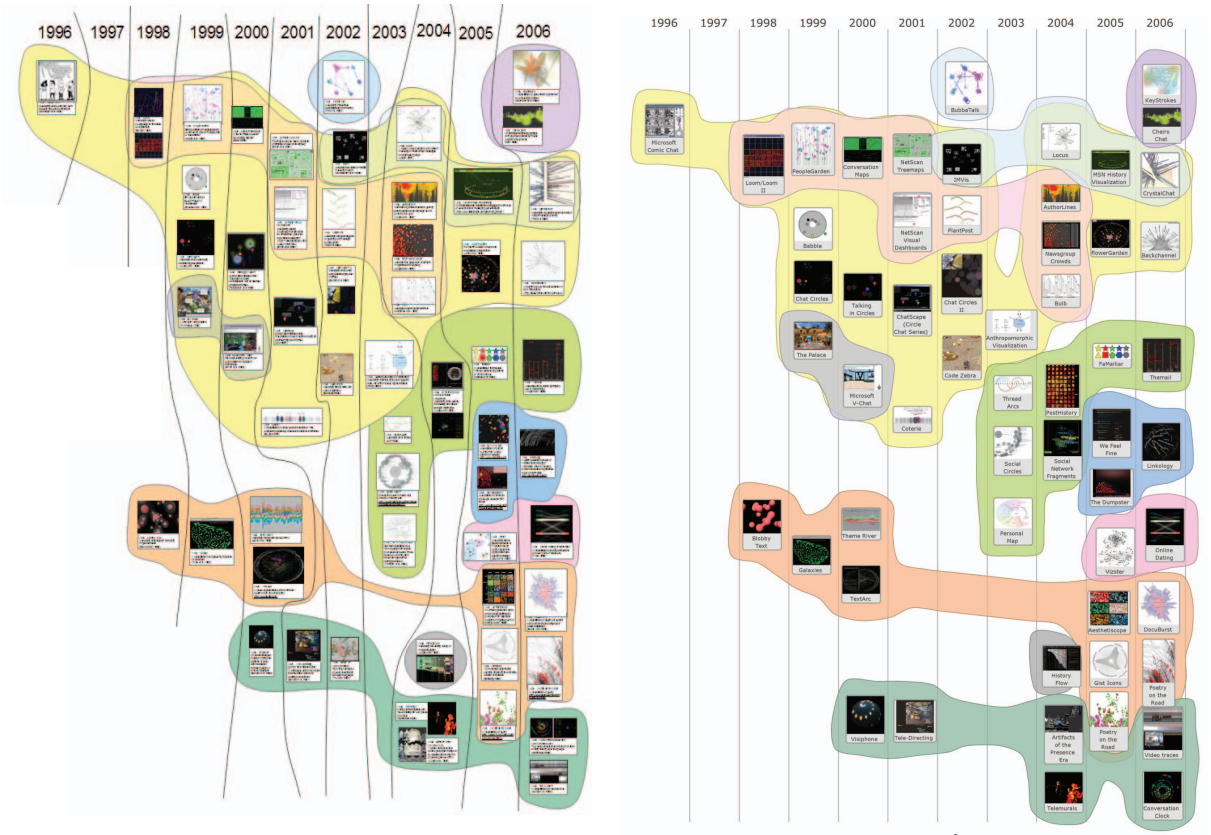

Fig. 7: Grouping research articles on a timeline. (a) Manually-created sketch (courtesy Tat [20]). (b) Bubble Sets visualization.

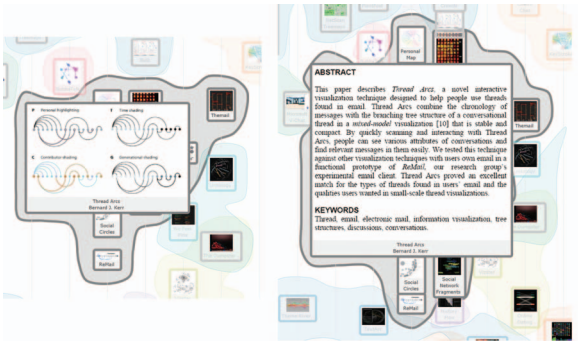

Fig. 8: Items can be expanded to reveal a larger image or the article's abstract. The boundary moves to accommodate the larger item, and other items move along the y-axis to remain visible and selectable.

Sets of hotels, subway entrances, and medical clinics (Figure 9) may help them find a hotel that is central to several medical clinics and near a subway entrance.

4.4 Sets over Scatterplots

Scatterplots have clearly defined spatiality due to the numerical positioning of items. We add Bubble Sets to a reimplementation of the well known GapMinder Trendalyzer [19]. This scatterplot shows fertility rate against life expectancy and is animated over time. Data points represent countries, sized by population. Colour (and set membership) is defined by the continent. The grouping of the sub-Saharan Africa countries, highlighted in Figure 10, reveals that while most of the countries in this set had high fertility rates and low life expectancies in 1985, there are two outliers, Mauritius and Reunion, which are islands in the Indian Ocean. As the data set includes data for many years, and since Bubbles Sets are calculated at interactive rates, the temporal changes can be convincingly shown through animation.

5 Discussion and Future Work

We have presented Bubble Sets, a method for automatically drawing set membership groups over existing visualizations with different degrees of requirements for primary spatial rights. In contrast to other overlaid containment set visualizations, Bubble Sets maximizes set membership inclusion and minimizes inclusion of non-set members. In fact, Bubble Sets can guarantee that all set members will be within one container, as opposed to the more common multiple disjoint containers. While Bubble Sets cannot guarantee non-set member exclusion, the routing algorithm minimizes these occurrences.

Within our isocontour approach we have implemented several heuristics to reduce surface calculation and rendering time, such as grouping pixels for potential calculations and restricting the regions in which items influence the potential field. The current implementation works without noticeable lag (items can be dragged and the surface follows) for our examples (order of 100 nodes, 10–20 sets). For example, it takes on average 105ms to calculate the virtual edge set, fill the energy field, find the contour, and render the Sub-Saharan Africa set in a window size 1920×1200 pixels. That set has 48 items and the entire scatter plot has 196 points. The majority of this time is spent creating the virtual edge set. An incremental approach, using A^* search as in [23] may provide improvements in speed and stability. As the number of items, the screen resolution, or the number of sets increases, so will the rendering time. Additional techniques, such as grouping close items into larger pseudo-nodes, and caching the energy field values between frames may increase the capacity of the system.

Acknowledgments

Thanks to the following organizations for supporting this research: iCORE, CFI, NSERC, and SMART Technologies.

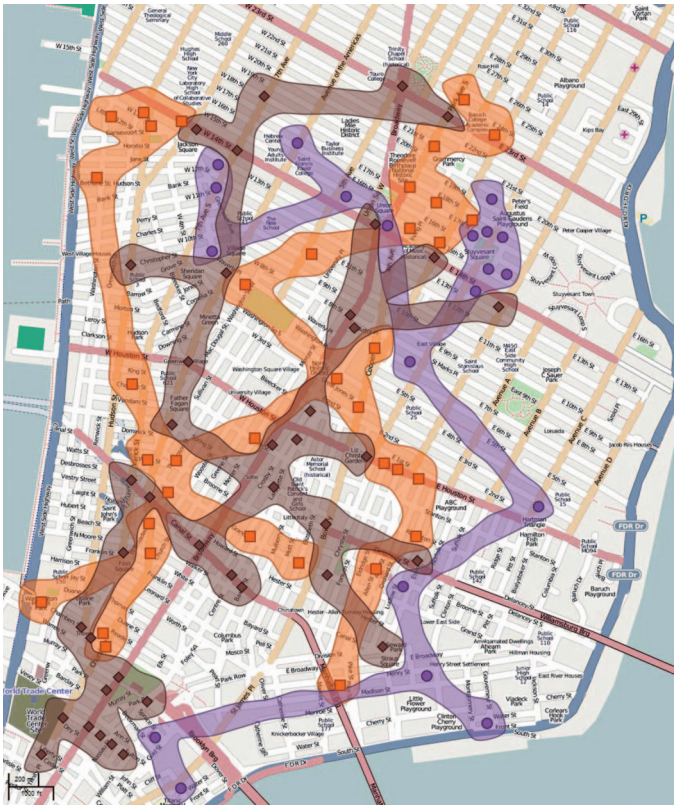

Fig. 9: Sets of geographically-defined items in lower Manhattan, showing hotels (orange), subway stations (brown), and medical clinics (purple). Medical clinics are noticably absent on the West side, and there is a cluster of clinics and hotels near transit in the Northeast corner.

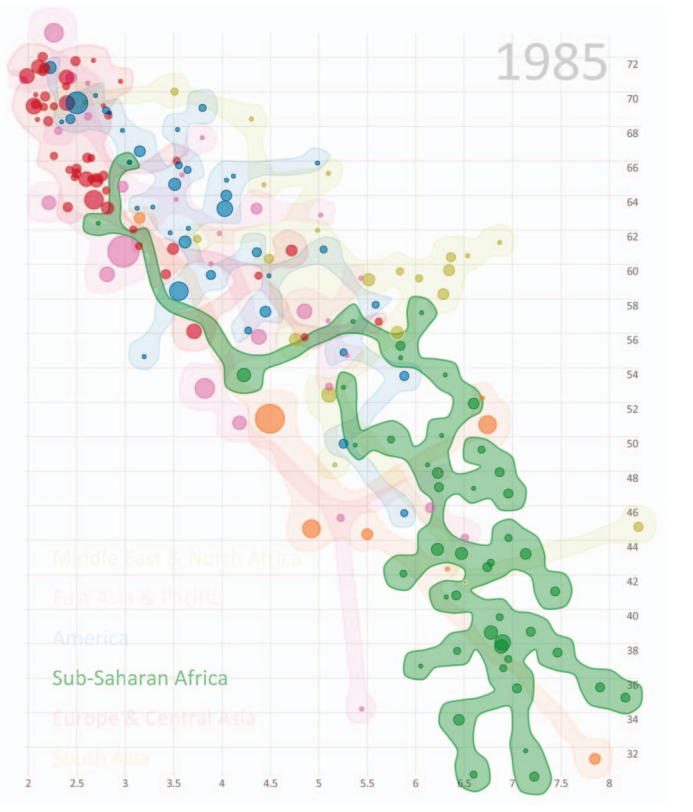

Fig. 10: A scatterplot of fertility rate by life expectancy by country. Hovering on a set member causes all non-members and other sets to be made transparent, clarifying set membership. Here, enclosure eases discovery of the outliers in the upper left, as well as giving a general impression about the spatial distribution of the set.

REFERENCES

[1] M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In *Proc. of the Asia-Pacific Symp. on Visualization*, pages 133–140, 2007.

[2] D. Bauer, P. Fastrez, and J. Hollan. Spatial tools for managing personal information collections. In *Proc. of the Hawaii Int. Conf. on System Sciences (HICSS)*, page 104.2, 2005.

[3] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983.

[4] J. F. Blinn. A generalization of albegraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[5] C. Buchheim, M. Jünger, and S. Leipert. Improving walker's algortihm to run in linear time. In *Proc. of the Int. Symp. on Graph Drawing*, number 2528 in LNCS, pages 344–353. Springer, 2002.

[6] H. Byelas and A. Telea. Visualization of areas of interest in software architecture diagrams. In *Proc. of SOFTVIS*, pages 105–114. ACM, 2006.

[7] C. Collins and S. Carpendale. VisLink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics (Proc. of the IEEE Conf. on Information Visualization)*, 13(6), 2007.

[8] T. Dwyer, K. Marriott, F. Schreiber, P. J. Stuckley, M. Woodward, and M. Wybrow. Exploration of networks using overview+detail with constraint-based cooperative layout. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1293–1300, 2008.

[9] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1141–1148, 2008.

[10] J.-D. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant. Overlaying graph links on Treemaps. In *Proc. of IEEE Symp. on Information Visualization, Poster Session*, pages 82–83, 2003.

[11] W. Freiler, K. Matković, and H. Hauser. Interactive visual analytics of set-typed data. *IEEE Transactions on Visualization and Computer Graphics (Proc. of the IEEE Conf. on Information Visualization)*, 14(6):1340–1347, 2008.

[12] J. Heer, S. K. Card, and J. A. Landay. prefuse: a toolkit for interactive information visualization. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. ACM Press, 2005.

[13] J. Heer and danah boyd. Vizster: Visualizing online social networks. In *Proc. of the IEEE Symp. on Information Visualization*, 2005.

[14] C. Heine and G. Scheuermann. Manual clustering refinement using interaction with blobs. In *Proc. of Eurographics/IEEE-VGTC Symp. on Visualization*. The Eurographics Association, 2007.

[15] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. of the Int. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 163–169, 1987.

[16] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *J. Visual Languages and Computing*, 6:183–210, 1995.

[17] A. Perer and B. Shneiderman. Balancing systematic and flexible exploration of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):693–700, 2006.

[18] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In *Proc. of the IEEE Symp. on Information Visualization*, pages 219–224, 2005.

[19] H. Rosling. Gapminder [online]. 2009 [cited 31 March, 2009]. Available from: `http://www.gapminder.org/`.

[20] A. Tat. Visualizing digital communication. Master's thesis, University of Calgary, 2007.

[21] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2nd edition, 2004.

[22] N. Watanabe, M. Washida, and T. Igarashi. Bubble clusters: An interface for manipulating spatial aggregation of graphical objects. In *Proc. of ACM Symp. on User Interface Software and Technology*. ACM, Oct. 2007.

[23] M. Wybrow, K. Marriott, and P. J. Stuckey. Incremental connector routing. In P. Healy and N. S. Nikolov, editors, *Proc. of the 13th Int. Symp. on Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 446–457. Springer, 2005.