1 Introduction

At any moment in time we are driven by, and are an integral part of, many interconnected and dynamically changing networks. Our species has evolved in the context of diverse ecological, biological, social, and other networks over thousands of years. We have created advanced environments in the shape of cities, water and power networks, streets, airline systems, and the Internet. Analyzing, exploring, and understanding these complex, interdependent, multi-level networks requires new, more efficient, and more intuitive graph analysis and visualization approaches [23, 95, 111].

Many algorithms, tools, and online services exist to analyze and visualize networks. However, few can be applied to study large-scale, multi-level networks, and the output of those are either hairball-like visualizations or counter-intuitive hierarchical cluster representations (via high-level meta-nodes and meta-edges). A recent data visualization literacy study by PI Börner *et al.* involving more than a thousand youth and adult visitors across five science museums in the U.S. revealed that most people cannot read such networks [132]. On the other hand, recent joint work by Kobourov and Börner, comparing standard network graph layouts and map-like visualizations of networks, shows that map-like visualization are superior in terms of task performance (time and accuracy) [109] and memorization and recall [107]; see Fig. 1.

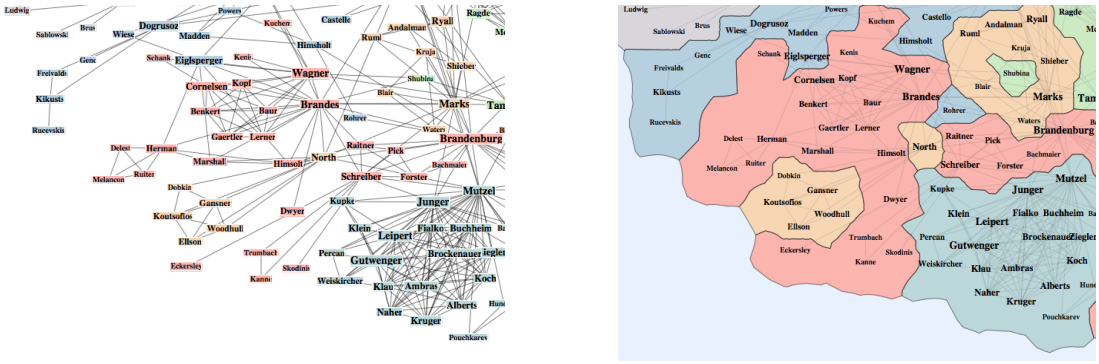


Figure 1: A traditional node-link visualization of a piece of a collaboration graph (left) and the map-like representation (right). Note that although the vertex placement and edge routes are the same, the map-like representation provides higher task accuracy and provides better recall of the data shown in the visualization.

Consequently, this proposal argues for algorithm development toward map-like visualizations of graphs. Specifically, we propose to develop a novel multi-level (i.e., multiple levels of detail) graph representation based on Multi-Level Graph Spanners (MLGS) that takes the results from [107, 109] and extends them to large real-world networks and a multi-level setting. The main difference of the proposed MLGS approach to existing approaches are:

- The MLGS approach displays real vertices from the underlying large graph at every level of detail.
- The MLGS approach displays real paths connecting the vertices at every level of detail.
- The MLGS approach clusters and embeds (i.e., lays out) the graph, providing a familiar map-like representation at every level, while also providing comparable information density at every level.

The proposed work aims to make large-scale, multi-level networks easy to explore, understand, and communicate by building on our ability to read, interact with, and navigate cartographic maps. More than 40 million people use Google Maps, Apple Maps, and other interactive visualizations to manage spatial tasks, such as zooming and panning, obtaining directions, and accessing details. When zooming in or out, the information density stays roughly the same; see example in Fig. 2 (left). At the macro level, the world and its seven continents are shown; zooming into North America and then the U.S. makes country and then state boundaries visible. Exploring the state level, we see major cities in each state and how they are connected by interstate routes. Zooming in further makes county boundaries, smaller cities, and roads between those cities visible.

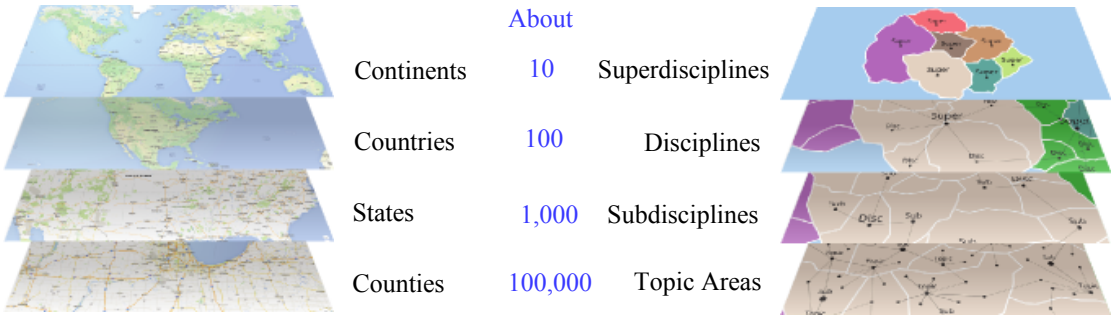


Figure 2: Multi-level map of the world (left) and a sketch of the envisioned multi-level, topical STH map (right).

The algorithm development proposed here is driven by the need to develop a more comprehensive classification system and to provide an easy to navigate map of all of science. The UCSD map of science and classification system was created by an international team using large scale datasets [20], see details in section 3. It is a two-level map that features 554 subdisciplines aggregated in to 13 disciplines of science (e.g., health professionals, brain research, social science). However, it is too coarse and aggregated to capture the structure and dynamics of science at level of topic areas. The MLGS approach proposed here will make it possible to effectively explore and communicate science across many more levels. Fig. 2 (right) shows a schematic example of the envisioned multi-level classification system and map of science, technology, and humanities (STH), subsequently called the STH map. The lowest cluster level shows approximately $100,000$ topical areas (teams of scholars and their publications) that group about 63M publications into coherent sets based on shared citations and/or keywords. The next higher level has about $1,000$ subdisciplines. Zooming out reveals about 100 disciplines (commonly known via department names or study program headings). The top level shows 10 superdisciplines (e.g., chemistry, math, physics, arts and humanities).

In line with the AitF solicitation, we will design and analyze provably efficient and provably accurate algorithms. These are embedded in the full software development lifecycle, starting with detailed user needs analysis and proceeding via algorithm design, implementation, deployment, testing and system optimization in an application domain. Algorithms developed and domain visualizations implemented in this project will be validated using quantitative and qualitative methods.

2 Multi-Level Graph Representation (Lead by Kobourov)

Here, we describe the MLGS approach and key research challenges, discuss related prior work, present the planned research, and outline the proposed quantitative validation.

2.1 MLGS Definition and Research Problem

The key research problem is the development of multi-level graph representation algorithms that provide the look and feel of a zoomable cartographic map. The underlying mathematical problem can be thought of computing a sequence of graphs that "approximate" the original graph at different levels of detail (LOD). In traditional LOD graph representations, the abstract graphs contain meta-nodes (representing a cluster of original vertices) and meta-edges (representing edges from some vertex in one cluster to some vertex in the other cluster). While this is a plausible solution for schematically showing hierarchical relationship, it does not lend itself to easy navigation, exploration, and other visual analytics tasks. In contrast, we propose a map-inspired, multi-level graph visualization that shows a nested hierarchy of graphs, each of which contains real vertices (from the original graph) and real paths connecting these vertices. This mimics standard geographic maps that show real cities and real roads at every LOD. However, unlike in geographic

maps, in our setting we have two additional competing objectives. On one hand, the graphs should be sparse enough to allow a legible visualization. On the other hand, the graphs should accurately describe the structure (e.g., connectivity, degree distribution, edge density) of the original graph.

To compute such a representation for a given graph G, we need a sequence of progressively coarser graphs $G_0 \supset G_1 \supset \cdots \supset G_L$, representing G on every LOD. Given our desire to use real vertices and real paths (rather than meta-vertices and meta-edges), we can further state that $V = V_0 \supset V_1 \supset \cdots \supset V_L$ and $E = E_0 \supset E_1 \supset \cdots \supset E_L$. In particular, for $0 < i \leq L$, the graph $G_i = (V_i, E_i)$ is defined so that its vertex set is a proper subset of the original vertex set ($V_i \subset V$) and its edge set corresponds to *paths* in the original graph: $e \in E_i \Rightarrow \exists p \in G$, and the path p is shown in such a way in the layout of G_i so as to make clear that there is a sequence of intermediate vertices that are not shown in the current level of detail (e.g., with a curve as opposed to a straight-line segment).

We next consider natural ways to create the sequence of graphs $G = G_0 \supset G_1 \supset \cdots \supset G_L$ and to use them to create a corresponding nested clustering and embedding/layout that provide map-like feel and interactions. In many domain applications (including ours), the underlying graph of interest is vertex-weighted and edge-weighted, with different weights encoding different types of information. For example, in the context of a collaboration graph, where the vertices are researchers and edges are collaborations, the vertex weight could be determined by the number of collaborations (vertex degree), by the H-index of the researcher, by the number of citations to papers by the researcher, etc. Similarly, the edge weight could be determined by the number of collaborations between the two corresponding researchers, by the structural role of the edge in the underlying graph (e.g., number of shortest paths that use this edge), etc. The underlying graph could also be unweighted, in which case we explicitly use structural properties to determine vertex and edge weights (e.g., degree centrality, betweenness centrality, eigenvector centrality).

We will then use the given or computed vertex weights, sort the vertices by non-increasing weight and create the vertex sets $V = V_0 \supset V_1 \supset \cdots \supset V_L$, such that V_1 removes roughly half of the vertices (the lighter ones), V_2 removes roughly half of the remaining vertices, and so on. This corresponds naturally to what we expect in maps: zooming out of a detailed map view, the smaller towns disappear leaving only the larger ones. But what about the edges in our graphs? When zooming out of a detailed map view, some roads disappear, but other roads remain and they do not get abstracted by directly connecting the remaining large cities. Similarly, we want the edges in our graphs to correspond to paths connecting our vertices (rather than abstract meta-edges).

2.2 Related Work

Prior work that is relevant to this proposal spans various topics such as algorithms for the layout of large graphs (where just one large layout is produced), algorithms for multi-level visualization of large graphs (where a hierarchy of graphs and meta-graphs is shown), as well as algorithms for Steiner trees and their variants, and graph spanners and their variants.

Large Graph Visualization: Most graph layout algorithms use either a force-directed [47, 64] or a stress model [28, 93] and provide a single static drawing. The force-directed model works well for small graphs, but does not scale for large networks. A popular speedup technique employs a multi-scale variant [65, 76] or the utilization of parallel computation as in VxOrd [25, 40]. GraphViz [48] uses a multi-scale force-directed method and the SFDP algorithm [79] that combines the multi-scale approach with a fast n-body simulation [7]. Stress minimization was introduced in the more general setting of multidimensional scaling (MDS) [94] and has been used to draw graphs as early as 1980 [114]. Simple stress functions can be efficiently optimized by exploiting fast algebraic operations. Specifically, the stress function can be globally minimized via majorization, which is guaranteed to converge. Modifications to the stress model include the strain model (classical scaling) [126], PivotMDS [28], COAST [71], and MaxEnt [72]. Although the above algorithms are fast and produce good drawings of regular grid-like graphs, the results are not as good for dense graphs, or small-world graphs [29].

Network layout algorithms are also provided in several libraries, such as GraphViz [48], OGDF [36], MSAGL [101], and VTK [112], which however, do not support interaction, navigation, and data manipulation. Visualization toolkits such as Prefuse [77], Tulip [6], Gephi [9], and yEd [129] support visual graph manipulation, and while they can handle large graphs, their rendering does not: even for graphs with a few thousand vertices, the amount of information rendered statically on the screen makes the visualization unusable.

There are research papers that describe interactive multi-level interfaces for exploring large graphs such as ASK-GraphView [1], topological fisheye views [69], and Grokker [105]. Software applications such as Pajek [41] for social networks, and Cytoscape [116] for biological data provide limited support for multi-level network visualization. These approaches rely on meta-graphs made out of meta-vertices and meta-edges, which make interactions such as semantic zooming, searching, and navigation very counter-intuitive.

Graph Spanners and Steiner Trees: The classic Steiner tree problem is one of Karp's original 21 NP-Complete problems [88]. A polynomial time approximation scheme exists if the underlying graph is planar [24], but not for general graphs [37]. More importantly, both the Steiner tree and metric Steiner tree (when edge weights satisfy the triangle inequality) can be efficiently approximated within a factor of $\ln 4 + e < 1.39$ [32]. The bottleneck Steiner tree, where the optimization is over the heaviest edge in the tree (rather than the sum of edges in tree), can be solved exactly in polynomial time [110], although the k-bottleneck variant (restricting the number of Steiner vertices) remains NP-hard [2]. In the vertex-weighted Steiner tree problem, the cost of the tree is determined by the weights of the vertices included in the solution (rather than the weights of the edges). The problem is NP-hard [99] but can be approximated within a logarithmic factor [75]. The Maximum Weight Connected Subgraph (MWCS) problem asks for a connected subgraph with maximum total weight in the given vertex-weighted graph. This problem is not as well-studied as the Steiner tree problem, but is also known to be NP-hard [86]. Note that none of these problems have been studied in a multi-level setting, which is a goal of this proposed research.

In graph simplification, the goal is to filter out as many edges as possible, while preserving various graph properties. Earlier work on graph simplification focuses on preservation of graph cuts [11], spectra [121, 122], connectivity [135], and paths between vertices [103, 113]. Graph spanners aim to remove edges while providing guarantees about the distances in the modified graph. Specifically, a *t-spanner* of a graph G is a subgraph S, such that the shortest distance between every pair of vertices in S is at most t times their distance in G. It is known that, for an n-vertex graph with non-negative edge weights, there exists a $(2t-1)$-spanner containing $O(n^{1+1/t})$ edges, and such a spanner can be computed efficiently [4, 5]. Note again that although problems such as graph sparsification and graph spanners are related, none of these problems has been studied in a multi-level setting, which is a key goal of the proposed research.

Related concepts have been studied in the more practical setting of network science, where a sparse subset of important edges is called a *backbone* [56, 74, 134]. There is no commonly accepted formal definition of a backbone [106], and existing algorithms use heuristics that try to filter out some vertices and edges based on some measure of *centrality* (e.g., degree, eigenvector, and betweenness centrality) [103, 113, 115] but provide no exact or even approximate guarantees.

2.3 Planned Research

Given a large graph $G = (V, E)$ we are interested in computing a sequence of graphs that "approximate" the original graph at different levels of detail. More concretely, we would like to compute a hierarchy of progressively coarser graphs, $G = G_0 = (V_0, E_0) \supset G_1 = (V_1, E_1) \supset \cdots \supset G_L = (V_L, E_L)$, each of which contains real vertices (from the original graph) and real paths connecting these vertices. Given our desire to use real vertices and real paths (rather than meta-vertices and meta-edges), we have that $V = V_0 \supset V_1 \supset \cdots \supset V_L$ and $E = E_0 \supset E_1 \supset \cdots \supset E_L$.

A Tree-Based Approach: In its simplest version, where we only want to ensure the connectivity of the graphs on every level, our problem resembles a multi-level Steiner tree computation. Given an edge-weighted graph $G = (V, E)$ and a subset of vertices $S \subset V$, the Steiner tree problem asks for the min-cost tree that spans S. However, in the algorithmic problem stated above, we want the max-cost tree, i.e., the tree that preserves the strongest edges. More importantly, we want to compute recursive, multi-level Steiner trees that span the vertices present in $G_0, G_1, \ldots, G_L$.

Since the classic (1-level) Steiner tree problem is NP-hard [88], the multi-level max-cost Steiner tree problem is likely also NP-hard. The 1-level Steiner tree problem can be approximated within a constant and our first research problem is to determine whether the multi-level max-cost Steiner tree problem can also be approximated within a constant. We can convert the max-cost spanning tree problem into an equivalent min-cost one by negating the edge weights. This technique has been used to compute max-cost spanning trees in general (although it cannot be used to compute longest paths). Then, the multi-level Steiner tree problem can be formalized as follows; Given an undirected edge weighted graph $G = (V, E)$ and a filtration of the vertices of $G : V = V_0 \supset V_1 \supset \cdots \supset V_L$, compute edges $E = E_0 \supset E_1 \supset \cdots \supset E_L$, such that each graph $G_i = (V_i, E_i)$, $0 < i <= L$ is a tree and minimize the cost:

$$\sum c(E_0) + \sum c(E_1) + \cdots + \sum c(E_L).$$

Note that there are many possible ways to define the cost of the edges $c(E_i)$ in the trees: sum of all the edges along each path connecting two vertices of the tree, square root of the sum of squares of the edges along each path, etc., all the way to the max cost of any edge along a path (bottleneck multi-level Steiner tree). For example, when the cost of edges $c(E_i)$ is just the sum of all the edges present in G_i we need to minimize a prorated sum of the edges in the underlying graph:

$$\sum_{e \in E_0} c(e) + \sum_{e \in E_1} c(e) + \cdots + \sum_{e \in E_L} c(e) = \sum_{e \in E_0, e \notin E_1} c(e) + \sum_{e \in E_1, e \notin E_2} 2c(e) + \cdots + \sum_{e \in E_L} (L+1)c(e).$$

A Graph Spanner Approach: Using trees to connect the vertices in $G_1, \ldots, G_L$ via multi-level Steiner trees might be too restrictive and not represent the underlying graph well (e.g., in terms of connectivity and distances). A different problem formulation we will consider is the multi-level graph spanner problem. Recall that a t-spanner of a graph G is a subgraph S, such that the shortest distance between every pair of vertices in S is at most t times their distance in G. A spanner subgraph approximately preserves shortest paths in the input graph.

We need to modify the spanner definition in two important ways: (1) to be able to deal with paths in G (rather than with single edges) and (2) to accommodate our required multi-level representation. We will consider a bottom-up approach for defining and building a multi-level graph spanner. Given the vertex sets $V_0 \supset V_1 \supset \ldots, \supset V_L$ we can define the multi-level t-spanner as a hierarchy of graphs $S_0 = (V_0, E_0), S_1 = (V_1, E_1), \ldots, S_L = (V_L, E_L)$ such that the shortest distance between every pair of vertices in S_i is at most $2^i t$ times their distance in G. Note that for $i = 0$ this makes S_0 exactly the t-spanner of the original graph G, whereas for $i > 0$ the distance guarantees provided in the spanner for level i are proportional to both t and the level of detail i.

From Graph Hierarchy to Multi-Level Representation: Starting with the tree-based or graph spanner approach discussed above for creating a graph hierarchy, we can further add a hierarchical clustering and generate layouts for interactive exploration and navigation, such as semantic zooming and panning. In particular, we can organize the vertices into a hierarchy of clusters, such that (a) each cluster is a subset of the vertices of G; (b) any two clusters in the same LOD are disjoint, and (c) each cluster in G_i is properly contained in one cluster in G_{i+1}. Clustering of this type can be computed by assigning level-i vertices to the cluster of the nearest level-$(i+1)$ vertex. The algorithmic challenges can be summarized as follows:

1. Consider ways to construct the hierarchy of graphs $G_0, G_1, \ldots, G_L$, such that each graph is connected and represents the underlying graph well (in terms of distances and/or structure).
2. Study the complexity of the underlying problem of computing multi-level max-cost Steiner trees using different cost measures.
3. Study the complexity of the underlying problem of computing multi-level spanners, providing different guarantees, such as distance preservation.
4. Design efficient exact/approximation algorithms for computing multi-level Steiner trees and spanners.
5. Design efficient algorithms for creating a hierarchical clustering that corresponds to the multi-level graph hierarchy.

Formally, given the hierarchy of graphs and clustering, we will design efficient algorithms for placing the vertices and routing the paths, so that the graphs at each level match as closely as possible the distances in the original graph.

Multi-Level Layout: There are several desirable properties for the layout of a hierarchy of graphs:

- The layout should capture the distances in the underlying graph (as much as possible);
- Labels for vertices at level i and higher should be readable (non-overlapping) in G_i
- Vertices should be well-separated in the layout (avoiding the "hairball" effect).

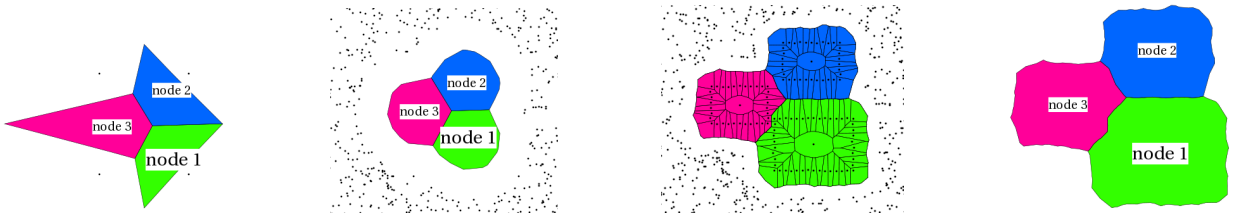
These properties are natural in graph visualization, but are novel in the context of multi-level visualization. The main aim of most graph-layout algorithms [66, 79, 93] is to preserve the underlying graph distances, while overlap removal techniques [46, 70] are used to make all vertex labels readable. However, we need to create a layout in which vertices *at every LOD* are well distributed and legible.

The two natural approaches for placing the vertices and routing the edges in our multi-level setting are top-down and bottom-up. In the bottom-up approach we can directly compute vertex positions and edge routes for the original graph $G = G_0$ and all coarser graphs inherit this information. When viewing graph G_i, for $0 < i \leq L$, we show the vertices and edges at level i or greater. Note that vertices at lower level might be completely invisible, or implicitly present as bends along paths connecting high-level vertices. This approach can leverage existing layout methods, but is limited by our ability to obtain high quality layouts of large graphs.

The top-down approach is more promising as we can efficiently create a high quality layout for the small graph on the highest level, G_L, and use it to efficiently create a high quality layout for the next graph in the hierarchy, G_{L-1}, and so on. In particular, suppose we place each vertex v of G_L in the plane in a way that provides enough space around it for all the vertices at lower levels that are clustered with v. Further, suppose edges in G_L are drawn with straight-line segments of length roughly proportional to the lengths of the corresponding paths in G. Then, we can obtain the layout for the next graph in the hierarchy, G_{L-1} from the layout of G_L using the initial placement of the vertices in V_L. For vertices in $V_{L-1} \setminus V_L$ we compute placement based on their distances from already placed higher-level vertices and information about the associated cluster size. A local refinement of vertex positions (including those that were fixed in the initial layout) follows. We proceed in this manner all the way to $G_0 = G$. A bottom-up propagation of vertex positions leads to the final layouts for all the graphs in the hierarchy. Note that edges (at higher LOD) are paths connecting pairs of vertices. As in geographic maps, exact details of the paths are not required and may distract a viewer. Hence, we will schematize the paths by reducing the number of internal points, e.g., using the Ramer-Douglas-Peucker algorithm [42, 104], which, given a curve composed of line segments, finds a similar curve with fewer points using dynamic programming.

We next discuss in more detail how we will provide the map-like look-and-feel of the multi-level representation.

Map-Like Visualization: Recall that our input graph has been processed to create a hierarchy of graphs $G = G_0 \supset G_1 \supset \cdots \supset G_L$ along with a hierarchical clustering. We will augment the layout of the hierarchies of graphs with map-like features, such as nested geographical regions that mimic continents,



(a) (b) (c) (d)

Figure 3: (a) Map from a Voronoi diagram of the vertices and corners of the bounding box. (b) Better construction of outer boundaries through placement of random points. (c) Voronoi diagram with additional points around the bounding boxes of the labels. (d) The final map.

which contain countries, which contain states, which contain counties, etc. There are several desirable properties:

- Every cluster should be represented by one simple and contiguous (connected) region in the plane.
- Every cluster should have a region in the plane proportional to its size (i.e., number of vertices).
- A region representing a subcluster should properly fit in the region allocated to its parent cluster.
- Regions should be as convex (or well-shaped) as possible.

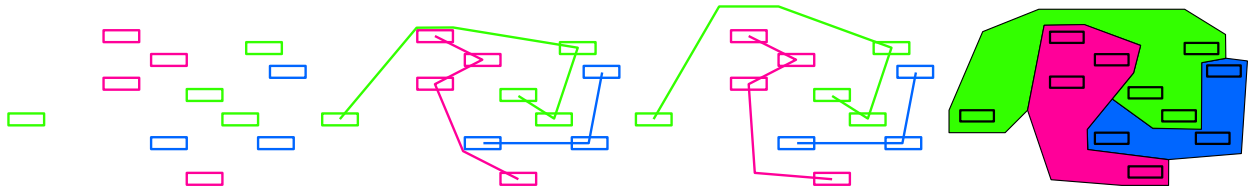
Once we have the layout and the clustering, we create a visualization that resembles a geographic map by using a modified Voronoi diagram of the vertices. We create "counties" from the lowest-level clusters, then create "states" from the next-level clusters, then "countries" from the next-level clusters, and so on; see Fig. 1. Unique colors are assigned, ensuring that no two adjacent regions have similar colors [81].

A simple way to obtain regions would be to create a Voronoi diagram of the vertices, together with four points on the four corners of the bounding box; see Fig. 3(a). A more map-like appearance can be obtained by adding a bit of noise, which results in less artificial borders; see Fig. 3(b). We will also address the practical problem of region sizes: small clusters (containing few vertices) should correspond to small regions, and large clusters should correspond to large regions. Voronoi cells of the same color (hence of the same cluster) will be merged to give the final map; see Fig. 3(d).

It is worth mentioning that the computed layout and the clustering constraints can potentially be at odds with each other. Nevertheless, we can guarantee that clusters are represented as contiguous regions (possibly at the expense of region-convexity) as follows. We begin by computing a crossing-free spanning tree of points belonging to a cluster. Once the tree is constructed, its vertices and edges become obstacles, and the procedure is repeated with the subsequent trees. Now we can grow contiguous, non-overlapping regions, starting from these disjoint trees. This procedure might result in "octopus"-like shapes that are neither aesthetically appealing nor practically useful for visualization. Hence, we require a method for creating regions that are as convex as possible. A possible approach addressing this problem (summarized in Fig. 4) is the following: *1. Tree construction:* compute spanning mutually non-crossing trees, connecting centers of rectangles corresponding to the same cluster, while minimizing the total edge lengths in the trees. *2. Force-directed adjustment:* modify the trees by adding buffers of free space around the segments of the trees, using a force-directed heuristic. *3. Create map regions:* use the modified trees to build contiguous, non-overlapping boundaries for all clusters and refine the boundaries as shown in Fig. 3.

2.4 Quantitative Validation

The quality of the basemap (showing the entire graph) can be evaluated in a quantitative fashion. There are several plausible measures, depending on what was optimized (e.g., distances when using graph spanners, costs when using Steiner trees, etc.) and on our interpretation of readability and usability of the visualization. For example, our recent study [89] indicates that the number of edge crossings—the most popular aesthetic



(a) Input (b) Tree Construction (c) Force-directed Adjustment (d) Computing Map Regions

Figure 4: Our method for creating contiguous regions for a clustered graph that optimizes convexity.

criteria for drawing graphs—may not be as important as previously believed, especially for larger graphs. On the other hand, it was experimentally observed that viewers prefer drawings with low "visual energy" of the layout [45]. As there are different formulations of such energy, we will evaluate several of the most promising ones. To this end, we assume that each of the proposed methods assigns point $p_v \in \mathbb{R}^2$ for every vertex $v \in V$ of graph G. We further assume that each edge $(u, v) \in E$ is assigned a weight $w(u, v) \in \mathbb{R}$. As suggested in [63], these weights are converted into ideal distances using a logarithmic transformation such as $d(u, v) = -\log[(1 - s) \cdot w(u, v) + s)]$, where $s > 0$. As examples, we describe three measures for evaluating the quality of the layout, *stress*, *distortion*, and *precision/recall*; see Fig. 5.

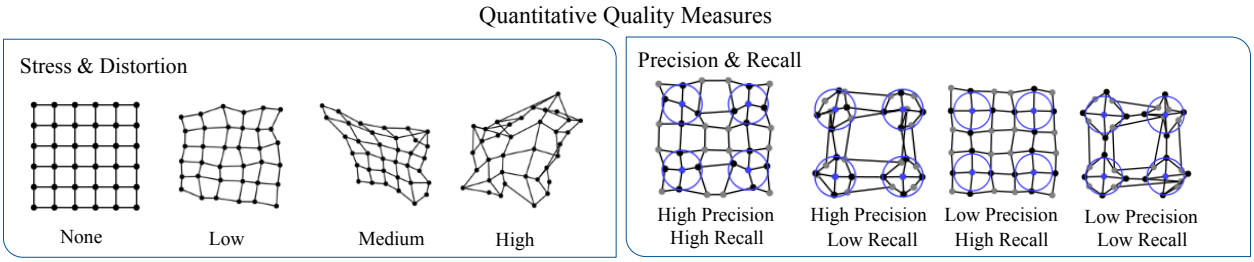


Figure 5: A 6×6 grid graph with varying levels of stress, distortion, and precision/recall.

Stress: The stress of a layout is the classic evaluation function used for MDS, which is based on the idea of modeling a graph as a system of charged vertices and springs connecting them [73]. The stress measures the energy of the spring system; we use the normalized version of the measure defined by

$$\frac{1}{m} \sum_{u,v \in V} w(u, v) \left(\frac{||p_u - p_v|| - d(u, v)}{\max(||p_u - p_v||, d(u, v))} \right)^2,$$

where $m = \frac{1}{2} \sum_{uv} w(u, v)$ and $||p_u - p_v||$ is the Euclidean distance between p_u and p_v. Low stress indicates a good solution.

Precision/Recall: Intuitively, precision and recall measure the rate of false positives (pairs of objects that have low similarity but are represented with small Euclidean distance in the map) and false negatives (pairs of objects that have high similarity but are represented with large Euclidean distance in the map) in the map. The goal of minimizing false positives and false negatives is similar to the goals of MDS [72, 94] and principal component analysis [87]. For a given k, the precision of point p_i in a layout is defined as the ratio of points within radius $r_i^k = ||p_i - p_i^k||$ (where p_i^k is the k-nearest neighbor of point p_i), which are neighbors of v_i. Conversely, the recall of point p_i is the ratio of points outside radius r_i^k that are not neighbors of v_i. The precision and recall of point set P is then the average of the precision and recall of each point $p_i \in P$.

Distortion: Distortion measures whether the distances between pairs of vertices are proportional to the desired distances. Consider a matrix of ideal distances with entry $d(u, v)$ for vertices u and v, and a matrix

of actual distances between corresponding points with $\Delta(u, v) = ||p_u - p_v||$. The matrices are seen as two random variables for which the correlation coefficient is computed. The value 1 indicates a perfect correspondence between ideal and desired distances, while 0 means that the distances are independent.

Note that the three measures above are examples of tangible layout quality metrics. We will customize the set of measures used depending on how the layout was obtained (e.g., via multi-level Steiner trees, or via multi-level graph spanners). We will further analyze which type of layout is better and which measures correlate with the quality of the layout with human subject studies in the application domain; see Sections 3.4 and 4.4.

3 Multi-Level Maps of Science, Technology, Humanities (Lead by Börner)

For centuries, cartographic maps have guided human exploration. Recent advances in data, algorithms, and computing infrastructures make it possible to map humankind's collective scholarly knowledge and technology expertise by using topic maps on which "continents" represent major areas of science (e.g., mathematics, physics, or medicine) and zooming reveals successively more detailed subareas, see Fig. 2 (right). Basemaps of science are generated by analyzing citations links between millions of publications and/or patents. "Data overlays" (e.g., showing all publications by one scholar, institution, or country) are generated by science-locating records based on topical similarity. Science maps are widely used to compare expertise profiles, to understand career trajectories, and to communicate emerging areas; see the special *PNAS* issue on *Mapping Knowledge Domains* [117], Börner's *Atlas* series [15, 17], with more than 1000 references to relevant work, and the *Places & Spaces: Mapping Science* exhibit [22].

However, many users have a hard time reading large-scale networks [132] and few can traverse or derive knowledge from multi-level presentations of networks. In January 2017, most maps of science, technology, or humanities support exactly one level of detail; very few support two (e.g., the UCSD map of science–the current de-facto science classification and mapping standard).

In this project, we will substantially increase the utility and readability of the UCSD science classification and mapping standard by computing a MLGS graph representation in support of multi-level zoom, exploration, and communication. Given results from our prior studies on the effectiveness and memorability of map-like visualization of large graphs [107, 109], we expect major improvements in terms of readability and utility. For the very first time, we will be able to offer a science look-up service that makes it easy for anyone to upload or select datasets to create data overlays, e.g., of expertise profiles of scholars/institutions/countries, career trajectories, or bursts of activity.

3.1 Key Research and Development Problems

Using the novel algorithms described in Section 2, together with a much larger publication dataset covering more than 100 years of publications (journals, conference proceedings, and books) the following research and development work will be performed:

1. Compute a new multi-level classification system and map;
2. Implement and deploy a science-coding service that accepts a set of text records and outputs their positions on the multi-level science map (analogous to a geo-coding service that accepts addresses and generates latitude-longitude coordinates);
3. Implement and make available online an interactive map-based visualization, providing easy exploration of multi-level science basemaps and data overlays.

All three parts and associated research problems and development challenges are discussed in Section 3.3.

3.2 Prior Work on Multi-Level Maps of Science

PI Börner's *Atlas of Science* features a 22-page timeline showing the evolution of science maps: from the first hand-rendered maps in the 1930s to interactive visualizations in 2007 [15]. Some maps are generated

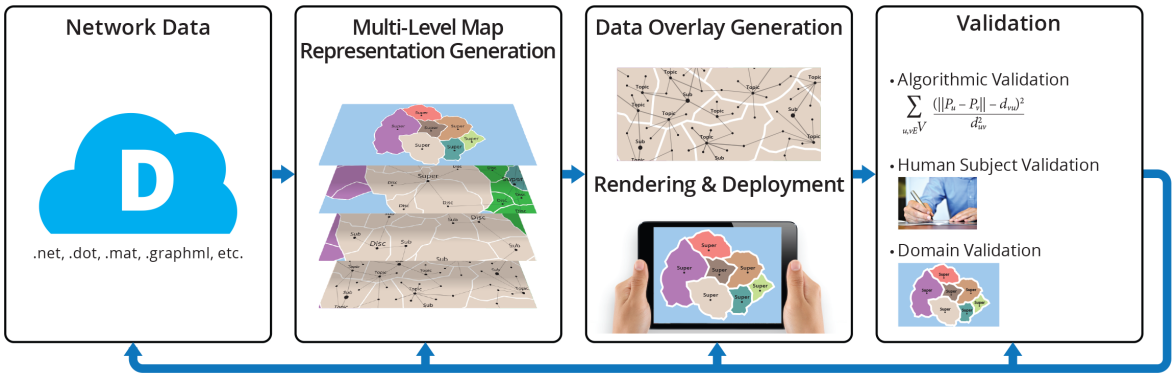


Figure 6: Iterative workflow: network data is read, MLGS representation is computed, basemap and data overlays are generated and deployed, results are validated.

manually: e.g., the *Map of Middle Earth* by Tolkien [125] or the *Map of Online Communities* by xkcd [100]. Here, we focus on data-driven map generation: e.g., the rendering of topical spaces based on co-occurring terms [33], work at PNNL visualization systems such as Themescape [130] and later Spire, In-Spire, or the more recent VOSviewer [127]. PI Kobourov's GMap approach [80] combines graph layout and graph clustering to render maps (e.g., of TV shows [68]) in support of visual exploration of topics in a particular conference or journal. The map of computer science (MoCS) [63] was generated from words and phrases extracted from over 2 million titles of research papers in the DBLP bibliography [97]. Skupin uses self-organizing maps linked with ESRI geographic information systems to render maps of textual documents [92, 118, 120]. Work by Waltman and Nees uses citation linkages to compute a publication-based classification system of science [128]. Today, the most comprehensive map of science and classification uses ten years of paper-level data from Thomson Reuters' Web of Science (WoS) and Elsevier's Scopus to group about 25,000 journals into 554 subdisciplines that are further aggregated into 13 disciplines; see data and detailed procedure in [20]. However, the two-level map of 13 disciplines and 554 subdisciplines is too coarse for organizing, navigating, managing, and making sense of millions of publications. Plus, none of the existing systems or maps supports the visualization of network clusters and backbones across multiple levels as depicted in Fig. 2 (right).

3.3 Planned Research and Development

This project will implement a fully functional online system for the interactive exploration of large, multi-level STH basemaps and associated data overlays. This free online service will provide a range of functions, including novel semantic zooming (for nodes/prototypes, links/paths/backbones, and groups/clusters), search (results are visualized with proportional symbol overlays and as heatmaps), and navigation (highlight shortest paths between pairs of nodes, and highlight accessible nodes from a given node). All source code and re-runnable workflows developed in this project will be made freely available for use in research, teaching, and practice. Leveraging the Network Workbench Tool (NWB) macroscope [16] with hundreds of thousands of users, we will provide a modular testbed for researchers to run, analyze, and validate new and existing algorithms on diverse datasets, thus lowering the data and code exchange barriers.

Hierarchical Classification System and STH Map: PI Börner's team is leading an international collaboration to update the global map of science and corresponding classification system standard. The 2012, most recent update of the UCSD classification system and map of science covers ten years (2001-2010) of Web of Science data and eight years (2001-2008) of Scopus data, see [20]. Using a combination of citation and term analysis, this basemap aggregates $25,000$ journals into 554 subdisciplines that are further aggregated into 13 scientific disciplines. The map is widely used in diverse desktop tools (e.g., the Sci2 Tool [124]) and online services such as the VIVO research networking system [19] or the interactive map of sustainability

research papers, patents, and funding [38]. Data overlays are created by matching publication records based on journal names or using keyword matches.

Computing Multi-Level Basemaps: The proposed map, called the STH map (Science, Technology, Humanities), will add more years as well as more data (technology and humanities), and it will support not only two but multiple levels thanks to algorithm development detailed in section 2; see sketch in Fig. 2 (right). It will be computed using IUNI Web of Science data [84] that covers 1900-2013 (54M papers and 1B references), book citation data from 2005-2013 (759k books and 30M references), and Proceedings data from 1990-2013 (8M papers and 82M references). We are in the process of acquiring the 2014-2016 data, which will further increase the number of papers/books and references. Using validated workflows to generate a weighted network of publications based on citation and topical information [20] and the MLGS algorithms developed in this project, we plan to compute an 8-level STH basemap with about 100M, 10M, 1M, 100k, 10k, 1k, 100, 10 clusters at subsequent levels.

Computing Data Overlays: The current UCSD map of science uses journal names to "science-locate" publications (i.e., publications are placed at the position of the journals in which they were published). It also provides keywords for each of the 554 subdisciplines to place patents, news, and other text records. Within this project, we will implement and compare different algorithms for the extraction and association of journals and keywords to clusters (regions) in the multi-level science map. This is analogous to geocoding that reads an address and returns latitude-longitude coordinates. This part of the project will benefit from a close collaboration with NIST; see *collaboration letter by Dr. Sriram*.

STH Lookup Service: The new STH basemap and data overlays will be used in a STH lock-up service that accepts a set of journal names or texts that are entered in form fields or uploaded as a file and generates a listing of associated positions for each level of the STH map of science. Summary statistics (e.g., number of unique positions per level), spread (as an indicator of interdisciplinarity), and distance to prototypes (as indicator for dominance) will be compiled as well.

Interactive Visualizations: Within this project, we will design and deploy an open-source system that reads large MLGS graphs, renders them as multi-level basemap reference systems, and generates interactive data overlays for exploration and communication. Default settings will produce a generic map with default clustering and layout, and advanced settings can be used to control the number of levels, the amount of node and edge filtration, the clustering and layout algorithms, map overlays, etc. The online system will be used for rapid workflow design (e.g., reading an EndNote or bibtex bibliography file, science-locating all papers on the STH basemap) and to run validation studies.

The IU team has developed OSGI/CIShell-compliant plug-and-play tools [16,83] for more than 15 years; recent work focuses on web service development for government and industry partners. PI Börner's team has extensive expertise creating visualizations using a variety of web technologies and libraries, including D3.js, Leaflet, Angular, and jQuery. Using the STH map plus a means to position records on the map (the STH lookup service), we will implement an online interactive map visualization that supports navigation of the STH map across multiple levels. Data overlays might show the expertise profiles of scholars, institutions, and entire countries; career trajectories of specific cohorts; bursts of activity; or emergent research areas.

Server-Client System Architecture: The proposed online system is novel as it requires multi-level semantic zoom and online interfaces that support the design of data analysis and visualization workflows requirements that lead to combining both server- and client-side rendering for different aspects of the visualizations; see Fig. 7. For example, cluster-analysis algorithms can be implemented as in-client operations, but this becomes increasingly untenable as datasets increase in size. By running these and other basemap-creation operations on a server or server farm, clients are allowed to devote their computing resources to displaying the results generated by these algorithms, data filtering, and layering. Utilizing the geographic

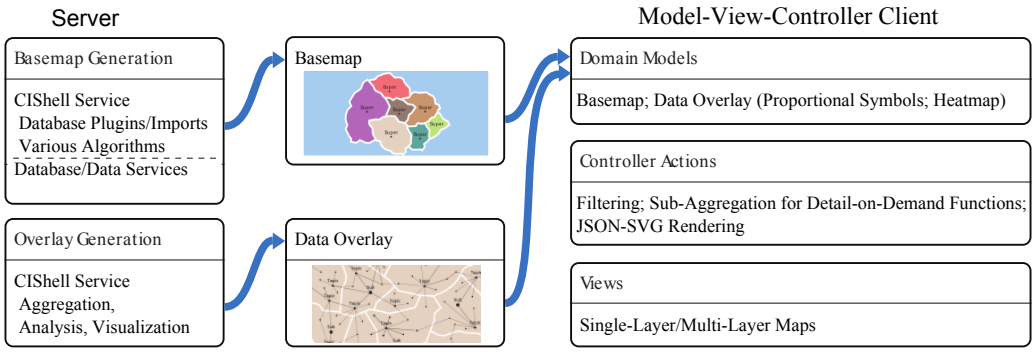


Figure 7: Server-client system architecture of the interactive, multi-level maps such as the STH mapping service.

map metaphor, we will show only a small portion of the input data: coarser maps at high level, with more details when zooming in. Hence, our system should be able to process large datasets and will not be limited by the power of a client-machine or client-browser. This part of the project will benefit from a close collaboration with ESRI; see collaboration letter by Dr. Wright.

Rendering Multi-Level Basemaps: The multi-level basemaps will be rendered using two approaches; results will be validated and the best solution will be used for the production service. (1) The first, more conventional approach uses rasterized tile-sets for consumption via geography-oriented engines such as Leaflet or Google Maps. This approach increases client-server bandwidth but also grants the benefit of caching rendered tiles on both server and client sides, meaning any client requesting the same underlying map may reuse the tiles generated for the first use. It also avoids the client-side computational expense for re-rendering the complete map for each pan or zoom operation. (2) The alternate approach produces underlying maps in vector format (e.g., SVG). Here, no additional bandwidth for rendering at different LODs is required, and in many cases, local GPUs can be used to assist rendering. (Note that server-side rendering is not a viable solution as it most easily lends itself to large sets of files that become static once rendered. Each client query against the underlying dataset has the potential to generate a new map, and storing all possible permutations of desired maps becomes an unreasonable solution. Client-side rendering is appropriate for adding data overlays, including additional tile sets, vector layers, and correlated metadata displays, to underlying server-generated maps. The state of in-browser rendering is improving, but complex compositing and vector overlays backed by large datasets might still easily overwhelm a client system, so care must be taken to use these judiciously.)

Rendering Data Overlays: Within this project, we will generate different data overlays, also called visual mappings, such as color coding (choropleth when using existing regions; heatmap when using data-defined regions) or proportional symbols that are easier to identify and compare (e.g., bars, circles, and squares). Heatmaps might be rendered as a separate tile set on the server. Highlighted selections of "cross-continental" links between nodes or proportional symbols denoting node values might be rendered in-browser as a vector/polygon set.

3.4 Validation

There are no prior studies that validate combinations of clustering and backbone identification or the combination of both with layout algorithms for large-scale, multi-level networks. However, there does exist prior work that informs the validation studies proposed here. Work by Jianu *et al.* [85] is relevant as it aimed to understand which data overlays on networks are most effective: node coloring, GMap [80], BubbleSets [39], or LineSets [3]. One of the main results was that contiguous cluster representation is important in tasks that involve group perception (i.e., cluster identification) and understanding.

Joint work by PIs Börner and Kobourov [109] evaluated three types of visualization that are characterized by a progressive increase of the amount of encoded information: node diagrams (e.g., point cloud vi-

sualization), node-link diagrams (e.g., typical network visualization) and node-link-group diagrams. Node-link-group diagrams are similar to the visualizations proposed here, except that they have only one level of detail. The study covered a wide spectrum of tasks, based on network visualization taxonomies [96] and a recent group-level taxonomy [108]. Our findings indicate that adding links, or links and group representations, does not negatively impact performance (time and accuracy) of node-based tasks. Similarly, adding group representations does not negatively impact the performance of network-based tasks. A follow-up paper has shown that map-based visualizations are also more memorable than traditional node-link diagrams [107]. PI Kobourov has also studied the readability of graph drawings [89], map-based visualizations [90], and semantic word cloud visualizations [8].

Prior work by PI Börner and colleagues validated different similarity measures [25] and compared text-based vs. linkage-based approaches [26]. Most relevant to this project, her team has aimed to understand the legibility of large-scale science maps. For example, a corpus of metadata from over two million MEDLINE documents (data available in [26]) was studied to assess the effectiveness of text-based similarity methods; five different methods were applied over either MeSH headings, titles and abstracts, or both to create nine different clustering solutions [123]. Börner and colleagues validated a large-format map of 2 million MEDLINE publications; the results demonstrate that biomedical experts can identify their very own area of research and circle and name other areas [119].

With a process similar to [119], we will validate science maps using real-world datasets and domain experts interested in answering real-world questions. Specifically, we plan to validate and compare engagement, legibility, memorability, and utility of different visualizations. Exemplary studies are discussed here, but further studies will be made feasible by the flexibility of the plug-and-play workflow design system setup discussed in Section 3.3. All studies will be theoretically grounded on the task typology by Brehmer and Munzner [30, 96], the visualization framework by Börner [17], and the group-task taxonomy [108]. That is, an agreed upon framework covering different insight needs, task types, graphic symbol types, and graphic variable types will be used to render the visualizations and to develop study instruments.

The design of efficient backbone-identification, clustering, and layout algorithms rests on a number of assumptions that need to be validated in human subject studies. Specifically, we will look for correlations between task performance and the algorithmic measures (stress, distortion, coverage). We will also compare concrete vs. abstract representations of prototype nodes; visualizations in which important nodes have more or less space (analogous to big cities); different projections (2d, sphere, cylinder layout); and different label densities and label distances.

Legibility. This set of studies will validate the readability of our network visualizations (e.g., can users find certain nodes, edges, paths, clusters, or specific zoom levels?) using timed tasks (e.g., finding the shortest path between two nodes and finding the average degree of a set of nodes), exploratory tasks (e.g., understanding what is in the data and where), and targeted tasks (e.g., identifying specific nodes that interlink clusters). The planned studies will aim to answer the following questions: (1) Which visual representations of backbones, clusters, and prototype nodes work best? (2) What label density is best? Do full labels have to be used or do the first n characters of a text string suffice? (3) Are straight lines or curved lines of edges easier to read and understand? (4) Are white spaces between clusters beneficial for recognition and memorization of cluster shapes and boundaries?

Memorability. We will evaluate how well participants can remember node and edge attributes (e.g., name, position, size, color) and network attributes (e.g., names and number of paths and clusters, overall structure and shape, number of levels) in visualizations in the short term (e.g., minutes) and in the long term (e.g., days and weeks).

Engagement. Visualizations do not exist in a vacuum. Frequently, they are surrounded by other static or interactive materials that might distract a reader from truly engaging with a visualization. We will design and perform a series of experiments, exposing participants to different visualizations in order to understand when and why people engage with the visualizations. For example, in one experimental setup, we will ask

users to sign a user-study consent form and ask them to wait in a room with three large-scale maps hanging on the otherwise empty walls; we will record how much time they spend with each visualization and also test the knowledge they acquired. An interview will be conducted to understand the impact of content, design, or other factors on their interest in (and engagement with) these visualizations.

Utility. Which network visualizations provide actionable insights and/or prompt more meaningful questions, or are otherwise useful for human decision-making? To facilitate this study type, we will invite experts to solve specific tasks. A follow-up interview will attempt to tease out whether the visualization led the participants to discover patterns and trends that they were not asked to find.

We will use standard human subject studies procedures: After giving consent, participants will (i) review and complete a pre-test questionnaire that collects demographic info (age, gender, native language, expertise), (ii) read an information sheet with basic information on the specific study, and (iii) perform specific tasks using different visualizations. The participants will also be given the opportunity to provide additional feedback via post-test questionnaires/interviews.

4 Intellectual Merit and Broader Impacts

Intellectual Merit: The work proposed here contributes to *graph algorithms* by studying novel graph representations based on multi-level graph spanner and Steiner tree problems, and by designing efficient exact and approximate algorithms for these problems. The proposed work will also contribute to *information visualization* with the implementation of these methods in a fully functional system for multi-level network visualization based on the familiar map metaphor. Finally, the proposed work will also contribute to *science mapping standards* by providing effective means to explore and use large-scale, multi-level science maps of our collective scholarly knowledge online. Specifically, the multi-level science classification system and mapping standard plus associated science lookup service will make it easy for anyone to create data overlays of expertise profiles of scholars/institutions/countries, career trajectories, or trends over time; see letter from *Clarivate Analytics (formerly Thomson Reuters)* declaring strong interest to collaborate on multi-level science mapping standards.

Broader Impacts: As with prior work by the PIs, source code and re-runnable workflows developed in this project will be made available for use in research and teaching. Moreover, functional tools and usable systems will also be made available to the general public. Continuing a tradition of involving students at all levels, this project will actively engage undergraduates, graduates and postgraduates. Interdisciplinary workshops will bring experts from graph algorithms, information visualization, and network science. Research papers, presentations, exhibitions, and tutorials will reach wide audiences, while new algorithms and visualizations will be disseminated using the Information Visualization MOOC that students from more than 100 countries take every year.

The algorithms developed in this project are generalizable to other domains where large graphs need to be visualized/explored/queried; see additional letters from: *National Institute of Standards and Technology (NIST)*, interested to "collaborate on algorithms, tools, online services, and standards development in the area of network science" and *ESRI* welcoming the opportunity to mirror or link content within the ArcGIS cloud-based platform that currently serves 500,000 users on a daily basis.

Education. Students at all levels (including undergraduates via REU supplements) will be involved in all phases of the proposed research. PIs Börner and Kobourov have many peer-reviewed publications with undergraduate RAs funded by REUs, some of whom have won national and international awards (e.g., CRA Outstanding Undergraduate RA, Churchill Scholarship). For example, PIs Börner and Kobourov just published a paper with a high school student, Scott Emmons, as the main author. Scott has already won many awards such as a National Merit Scholarship, Honors Carolina, and the Robertson Scholarship providing unique "dual citizenship" at UNC and Duke University. The PIs will continue to engage students at all levels in the proposed work. Börner is an active member of the Women in Computing program at IU where

she is supervising several female PhD students. If funded, this project will directly support female PhD students at both IU and UA.

Outreach. Börner co-organized the NSF-Agenda Setting Conference: *Modelling Science, Technology, and Innovation* (http://modsti.cns.iu.edu) in 2016 and she co-organizes a Sackler Colloquium funded by the National Academies of Sciences on *Modelling and Visualizing Science and Technology Developments* that will take place in Dec. 2017. She curates the *Places & Spaces: Mapping Science* exhibit (http://scimaps.org) featuring 100 maps of science and 12 macroscopes by 226 authors, displayed at 332 venues, in 28 countries, on six continents. Science classification and mapping standards are central to these efforts.

Kobourov co-organized a Dagstuhl exhibition "Bending Reality: Where Arc and Science Meet." The PIs plan to organize a Dagstuhl seminar and exhibition that will bring together experts in graph algorithms and network science to discuss, advance, and disseminate the work proposed here.

5 Results from Prior NSF Support

Börner brings extensive expertise in the development of data analysis and visualization techniques for information access, understanding, and management. She is particularly interested in the study of the structure and evolution of scientific disciplines; the analysis and visualization of online activity; and the development of cyberinfrastructures for large-scale scientific collaboration and computation. Börner was PI of *Pathways: Sense-Making of Big Data, NSF ISE DRL-1223698 project, 2012-15, $250,000.* The project aimed to create a foundation for the design of museum exhibits and educational programs that teach museum visitors how to explore, engage, and make better sense of big data. Specifically, 1120 visitors across five U.S. science museums were interviewed to understand if and how visitors read data visualizations. We were able to document dominant mental structures used by people to organize data; how initial interpretations of a visualization are maintained, even when probing for deeper analysis; and that meaning-making is more successful when constructing complex visualizations rather than deconstructing them. **Intellectual Merit:** A theoretical visualization framework was developed and used for the classification and design of data visualizations and the development of data visualization construction/deconstruction studies [12, 17, 18, 21, 78, 131–133]. **Broader Impacts:** Results have implications for how information visualizations are taught and used in formal and informal education, the media, or in relevant professions.

Kobourov is an expert in the development of algorithms for visualization of relational data. He is particularly interested in the design and analysis of geometric and graph algorithms, but also works on algorithm engineering, graph theory, and human-computer interaction. *CAREER: Embedding, Morphing, and Visualizing Dynamic Graphs, NSF-CCF-0545743, 2006-11, $419,645.* **Intellectual Merit:** The goal of this project was to develop practical algorithms for modeling and visualizing dynamic processes based on solid theoretical foundations. Efficient new algorithms for morphing in Euclidean and Riemannian spaces were designed and implemented. A new method for generalizing the notion of barycenter coordinates to non-Euclidean spaces was used to visualize dynamic graphs and was applied to applications such as mobile sensor localization and map building. **Broader Impacts:** This project led to many publications on visualization of large graphs [10, 31, 60] and graphs that evolve through time [82, 98]; software systems, including MoCS [63], GMap [80], GraphSET [53], and Lombardi [35]; book chapters on force-directed algorithms [91], simultaneous embedding [14], and map-based visualization [67]; new graph visualization models such as Lombardi graph drawing [43, 44, 102]; and two PhD theses (by Joe Fowler and Alex Estrella-Balderrama). Joe Fowler co-authored 8 papers [27, 50–52, 57–59, 62] and completed a PhD in 2009 [61]. Alejandro Estrella-Balderrama co-authored 8 papers [13, 34, 50–55] and completed a PhD in 2009 [49]. Undergraduates involved in this project won several awards: Gary Yee (CRA Outstanding Undergraduate Award), David Forester (Outstanding Computer Science Senior, UA), Katie Cunningham (Outstanding Senior College of Science, UA).