Map Visualizations for Graphs with Group Restrictions

Md Iqbal Hossain
University of Arizona
Tucson, Arizona, USA
hossain@arizona.edu

Ehsan Moradi[*]
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
e.moradi@usask.ca

Debajyoti Mondal
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
d.mondal@usask.ca

Stephen Kobourov
Technical University Munich
Munich, Bavaria, Germany
stephen.kobourov@tum.de

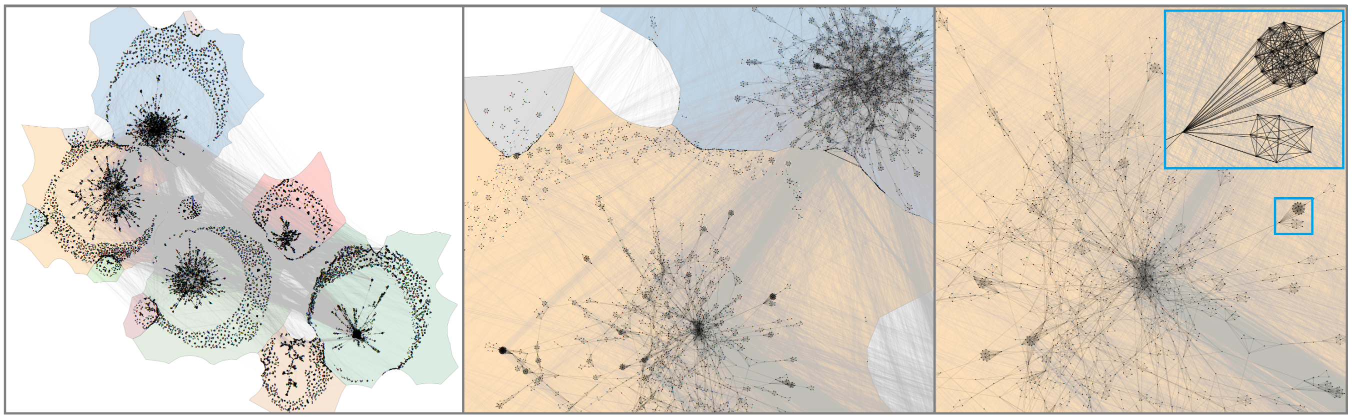


Figure 1: Left: A map-like visualization created by our algorithm for a co-authorship network with 26,325 nodes and 120,879 edges. Each conference is shown using a distinct country-like polygon with area proportional to its group size. The connections within group and between groups are in black and gray, respectively. Middle: A zoomed in view. Two dense subgraphs came close together due to their high connectivity. Sometimes nodes moved closer to polygon boundaries due to external connections, but remained inside their designated polygons. Right: A further zoomed-in view and a magnified view of a small subgraph.

ABSTRACT

A map visualization of a graph consists of a node-link diagram in which groups of nodes are enclosed in one or more polygonal regions, similar to countries in a geographic map. Many real-world graphs have naturally defined groups, e.g., a graph that represents collaborations between faculty members within a university, where the departments are the groups. A good visualization of such a graph should place departments that collaborate frequently as adjacent or nearby groups. While some set visualization methods can be used to create map visualizations for graphs with groups, the results can be poor and difficult to read due to fragmented groups or complicated polygonal shapes of the enclosing regions. With this in mind, we propose a new approach that constructs the polygons first and then renders the graph to obtain better control over the drawing properties. We design two methods based on this new approach and compare them with three prior techniques using seven quantitative metrics on several real-world datasets. Our experimental results demonstrate the proposed methods to outperform prior techniques in capturing the intended drawing features and have good performance in most of the metrics.

CCS CONCEPTS

• **Human-centered computing → Graph drawings**; **Visualization design and evaluation methods**.

KEYWORDS

Network Visualization, Force-Layout, Geographic Map Metaphor

ACM Reference Format:
Md Iqbal Hossain, Ehsan Moradi, Debajyoti Mondal, and Stephen Kobourov. 2025. Map Visualizations for Graphs with Group Restrictions. In *Proceedings of Graphics Interface 2025 (GI'25)*. ACM, New York, NY, USA, 12 pages.

[*]The author made substantial contributions to both algorithm development and experimental analysis as part of his PhD research.

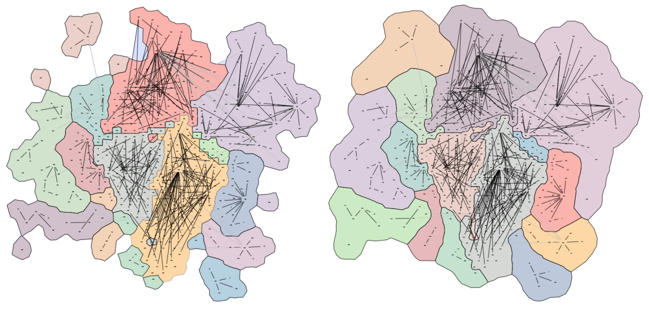


Figure 2: Left: A GMap visualization of the graph Bookland [21] with 499 nodes and 408 edges. Although there are 15 groups, but the visualization has over 25 polygons that represent these groups. Right: A MapSet visualization of the same graph.

1 INTRODUCTION

The popularity of geographic maps and our familiarity with them have made map-based visualization an attractive option even for visualizing non-geographic datasets [13]. In this paper we focus on the node-link visualization of graph data, which models relationships among a set of entities. GMap [15] is widely recognized for popularizing the use of geographic metaphors to visualize clusters in graphs. It creates a map-like representation, where the node-link layout is overlaid on a map, and the nodes of each cluster in the graph are enclosed with country-like polygonal regions.

To improve readability, a desirable property of such visualizations is the use of almost convex polygonal shapes. At the same time, their boundaries should remain irregular to align with the original design philosophy of resembling the boundaries of countries. However, since the algorithms that detect clusters [10] operate independently of the algorithm that generates the node-link layout, some nodes within a cluster may be positioned far apart in the visualization. Consequently, enclosing the nodes of each cluster using convex polygons may require multiple polygons per cluster, leading to a country-like representation with many fragmented regions per cluster (Figure 2 (Left)).

The representation with multiple polygons per cluster makes it harder for users to identify individual clusters and compare cluster sizes. The desire to keep clusters or groups of nodes (e.g., created from category-based partitions) together in the visualization gave rise to the concept of a group-in-a-box (GIB) layout [32, 35]. It first creates a treemap visualization with cells corresponding to the group sizes, and then places the subgraphs inside their corresponding cells. Although the rectangular cells of the treemap could be thought of as countries, this does not provide a geographic map-like aesthetic. To create a GIB layout using the geographic map metaphor, one may think of first creating a GMap visualization and then enclosing the nodes of each cluster in a single polygonal region. This problem is similar to a set visualization problem that given a collection of sets over some points (i.e., nodes in our setting) in the plane, seeks a way to create corresponding polygonal regions such that each region encloses the points of the set it represents. There are techniques to create such polygonal regions that either keep the polygons disjoint or allow for overlaps, e.g., MapSet [7], BubbleSets [4], LineSets [1] and other variants [23, 38, 41]. However, since the nodes in a group may spread across the whole layout, such polygonal regions are likely to be meandering polygons with high complexity and thus likely to deviate significantly from a natural-looking country shape (Figure 2(Right)).

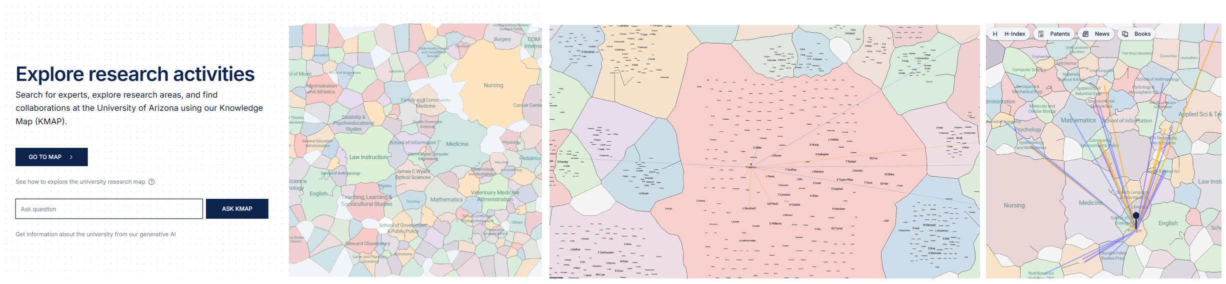
1.1 Motivation

While much effort has gone into creating map visualizations that enclose each cluster within a single polygonal region, little focus has been placed on creating such visualizations to capture cluster sizes and the relationships between clusters. In this paper we are interested in creating map visualization for graphs with predefined groups where groups must be represented by disjoint polygonal regions[1] with one polygon per group (Figure 1). Throughout the paper we will refer to such a visualization as **MVG (map visualization with group restriction)**. Some desired properties for MVG are as follows. The polygons should have a natural country-like shape (e.g., a blob-like interior that is not too narrow or spiraling) and the area of each polygon should be proportional to the group size. This helps estimate cluster sizes while adhering to country-like appeal. Both the relations among the nodes within each group and the relations between groups should be reflected by the relative proximity of their representative geometric elements (points or polygons) as much as possible, which helps understand graph structure. Additionally, the nodes should be distributed inside each polygon to utilize the space available, which enhances readability.

Consider the faculty collaboration graph at a university as a motivating example for the use of an MVG. In this graph, edges represent co-authorships or shared grants, and groups are defined by faculty members within individual departments. An MVG for such a graph provides Deans, Department Heads, and research administrators an understanding of the university's research ecosystem. It can reveal the impact of a department within the university and highlight the potential for forming interdisciplinary teams to apply for funding. This can be an effective approach to visualize graph data as the funding agencies worldwide are increasingly encouraging interdisciplinary projects. Furthermore, an MVG can identify individuals whose collaborations extend more widely outside their department than within. This insight can help faculty members establish new connections or reach out to colleagues across the university by exploring common collaborators.

Existing techniques, such as GMap [15] and set visualization methods [1, 4, 7], are not suitable for constructing MVGs. These approaches depend on the overall graph layout and fail to explicitly account for relationships among predefined groups. Closest to our work is CBA [21] which ensures that predefined groups must be enclosed inside disjoint polygonal regions. This approach first lays out the graph, next allocates disjoint square regions for each group based on the barycenter of the nodes in the group, then fits the drawing of each group into the allocated square, and finally, creates polygons and deforms them using a force-based layout [36] allowing nodes to be pulled towards their original location. Although this approach addresses the MVG drawing problem, it relies more

[1]Here two regions can be adjacent, i.e., share part of their boundaries, but the interior of the polygons must be non-overlapping.


Explore research activities

Search for experts, explore research areas, and find collaborations at the University of Arizona using our Knowledge Map (KMAP).

See how to explore the university research map

Get information about the university from our generative AI

Figure 3: Left: KMap, an MVG to visualize faculty collaborations. Middle-Right: Two views when a person has been selected.

heavily on the whole graph layout than the prescribed group information and the deformation of the polygons changes the polygon area. The lack of research on whether one could use the group information to first create the polygons and then distribute the nodes inside them motivates the technical contribution of this paper.

1.2 Contributions

Our primary contribution is to present two novel approaches for constructing MVGs, develop metrics to evaluate their structural properties, and demonstrate the superiority of our methods over prior techniques through an empirical analysis. In contrast to previous methods, we build MVG by first creating polygons for the groups and then distributing the nodes of each group into its polygonal region. Since the polygons are computed beforehand, this gives us better control over the polygon shapes. Our secondary contribution is to build a running system, where an MVG is being used within the University of Arizona as the 'Institutional Knowledge Map' (KMap) [19]. Figure 3 shows a snapshot of the MVG showing the departments and two views where a person has been selected. In this paper we focus on the algorithms that we developed for creating MVG, and compare and contrast these algorithms with the existing methods in the literature using quantitative evaluation. However, we envision a full-fledged evaluation of the KMap system with end users and domain experts in the future.

We evaluate our proposed methods with three existing approaches from the literature using benchmark datasets [21], as well as using real-life datasets from DBLP [5] and university collaboration. We use seven quantitative metrics to compare the desired structural properties (five new metrics and two existing ones, i.e., stress and edge crossing). The experimental results show our methods are the first of their kind to produce visually appealing layouts for large graphs, outperforming existing methods that often result in unreadable designs. Both approaches perform well across most metrics. While the first method is conceptually straightforward, the second method delivers better layouts with fewer edge crossings.

2 RELATED WORK

In this section we discuss the research related to cluster visualization and techniques that draw graphs as geographic maps.

Spatial metaphor is often used to visualize non-geometric data. A classic example of this is self-organizing maps (SOMs) to visualize documents [37]. The spatializations of documents present them metaphorically as points where similar documents lie in a homogeneous thematic region. The popularity of using spatial metaphors stems from our intuitive understanding of geographic space [3, 40]. Such visualization of maps can be found to display similarities among scientific disciplines [2], topics in conferences [11], software data [25], and so on. Our approach to designing MVG follows the same design principle of employing a spatial metaphor.

There is a rich body of literature on visualizing group structures in graphs [39]. GMap is one of the earliest works for representing graphs as geographic maps. GMap was originally proposed to visualize recommendations where the visualization creates a layout of graphs and encloses graph clusters inside colored polygonal regions. A number of usages of GMap have appeared in the literature since then for producing visualizations for more general data [14]. We will use GMap as a subroutine to generate country-like shapes, while our contribution will focus on improved organization of the countries and better design of the node-link diagram within the countries to reveal graph structure.

Past user studies show GMap to be faster for search and exploration tasks compared to other counterparts such as BubbleSets and LineSets [17]. However, GMap sometimes creates fragmented polygons for a single cluster which makes it slow for group-based tasks. Visualization of point sets on a geographic map is another line of work that relates to our context. Given several sets of points on a geographic map, a common way to represent them so that they are easier to identify or separate visually is to enclose or connect the members of each set using geometric objects (e.g., polygon or line). Such representations are commonly known as set visualizations. These objects are generally allowed to intersect. Examples of such visualizations are BubbleSets [4], LineSets [1], Euler diagrams [34], etc. These methods can be adapted to create MVGs but they may create overlapping and meandering polygonal regions as these algorithms do not consider the underlying graph relation into account. The user preferences to have a single polygon per cluster motivated the study of creating contiguous map visualizations where polygons are non-overlapping and each cluster corresponds to a single polygon [21]. Our algorithm also ensures single polygon per group. Unlike previous algorithms that generate polygons based on node-link diagrams, our approach constructs node-link diagrams based on polygon organization. This methodology provides enhanced control for achieving the desired MVG properties.

There are several techniques for visualizing large graphs, either through hierarchical layouts or using map-like zoomable interfaces [24, 29, 30]. GraphMaps [26, 30] proposes to reduce clutter by distributing nodes to different zoom levels and routing edges on shared curves. Cornac is another system that uses such layered techniques to visualize graphs with millions of nodes and edges [33]. Although these techniques are for creating scalable node-link diagrams and do not leverage the country-like metaphor, we draw inspiration from these studies when implementing MVG for practical applications [19] that involve large graphs.

3 EXISTING VISUALIZATION METHODS

We first discuss GMap [15] and MapSet [7]. Next, we review the method CBA [21] to create MVG. Finally, we describe our methods for creating MVG in Section 4.

3.1 GMap

The GMap approach [15] creates a layout of the whole graph created using force-based layout [12], then creates polygons to enclose the nodes of a group, and finally, attempts to enclose the nodes of a group in a polygonal region. However, this method may generate more polygons per group. Given a graph G, GMap first creates a drawing D with an existing layout algorithm such as force-based layouts [20] which starts with a random placement for the nodes and iteratively moves them by simulating repulsion and attraction forces to avoid node overlap and bring adjacent nodes closer to each other. GMap then generates the countries by partitioning the plane. Specifically, the region corresponding to a country is computed based on a Voronoi diagram where the cells for the nodes that belong to the same cluster are merged. When constructing Voronoi diagram sites, GMap adds some random points in the unbounded space of the layout and some points on the bounding box of the nodes to avoid sharp corners in the polygons and to give the outer boundary of the drawing a natural-looking shape.

3.2 MapSet

MapSet [7] also starts with a layout of the whole graph but guarantees that the nodes of a group are enclosed inside a single polygon. Here the polygons may have narrow extensions.
GMap and MapSet both have two major problems. First, since a force-based layout does not use the input group information, nodes can move freely and thus enclosing the nodes for each group inside a polygon often creates narrow and curved polygons. Second, the area of a polygon does not correspond to the size of the group it represents. To overcome this problem, CBA [21] proposes the scale-and-fit approach.

3.3 Scale-And-Fit Approach (SF-CBA)

CBA [21] starts with a force-based layout, and then computes the average coordinate of the nodes in each cluster. These coordinates are used to place a square for each cluster where the size of the square is set proportional to the size of each country. These squares may create overlaps. Therefore, the next step is to remove the edge overlaps using existing overlap removal algorithms [6]. Now that each country has a non-overlapping square-region reserved for its nodes, the nodes for each group are repositioned inside the corresponding square by scaling so that it fits the square. Finally, the polygons are created using the GMap approach and the nodes are pulled toward their original positions ensuring they remain inside the polygon boundaries whereas the polygon also deforms during this process. This last step is done leveraging the force-based algorithm ImPrEd [36] that allows for specifying edges that must not be crossed (i.e., the polygon edges). Furthermore, the edges can be marked as flexible so that they can expand or contract (by adding bend points) to accommodate the moving nodes.

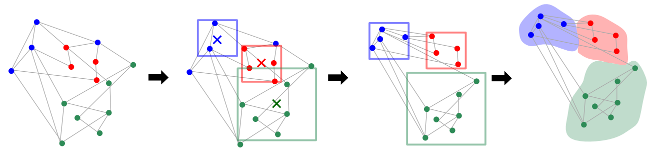


Figure 4: A conceptual representation for the steps of Scale and Fit Approach (SF-CBA). The initial square centers are marked in cross. The squares are moved to avoid overlap and the subgraph for each square is scaled to fit within it. Finally, GMap is used to create countries.

CBA attempts to attain low stress (see Section 5.3) by moving the nodes to the original position and thus deforming the polygon. An implementation of ImPrEd is not available to replicate the results on CBA. However, the authors in CBA [21] observed the quantitative performance of CBA is very similar to GMap, specifically when the underlying graph is dense. This indicates the polygons do not have much space to have significant deformation. In general such polygon deformation does not control the polygon area, and thus loses one of the key aspects (i.e., polygons are proportitional to the group sizes) that we want to preserve when building MVG. Consequently, we compare our methods with GMap and a version of CBA without the last step that uses ImPrEd. Specifically, we create a square for each group in a similar way as that of CBA, then the drawing of each group is scaled to fit its square, and finally, the polygons are created using the GMap approach. Figure 4 illustrates the process. Throughout the paper we refer to this method as SF-CBA.

4 OUR METHODS FOR CREATING MVGS

We propose two methods to create MVG. The first method — KMap — creates a weighted supergraph where each supernode corresponds to a square of size proportional to its group size and edges correspond to the edges between groups. Then a force-layout is used to place the squares [18]. Next, the subgraph of each group is drawn with a force-based layout and fit into its corresponding square. The final polygons are then created using the GMap approach. The second method — polygon constraint layout (PCL) — replaces the scale and fit approach with a force layout based node distribution inside each polygon created in KMap, where the nodes are constrained to be inside their corresponding polygons. Here we design novel force equations to optimize layout properties, which we consider to be a significant contribution for embedding graphs within complex polygonal shapes while accounting for their between-polygon relationships.

4.1 Method 1 (KMap)

This method is currently being used in our KMap system [19], which combines the following straightforward steps (Figure 5(left)).

Step 1: Given a graph, construct a weighted supergraph or group-group network G_c, where each supernode represents a group from the original graph G, and each edge represents a connection between two groups. The supernode weight in G_c reflects the number of nodes in the corresponding group of G, while the edge weight indicates the strength of the connection between two groups. We then use a force layout to create an embedding of the rectangles such that supernodes are drawn as squares with sizes proportional to their weights and the squares do not overlap.

Step 2: The subgraph induced by a group is drawn separately using a force layout and scaled to fit within its square.

Step 3: Use the node coordinates, group information and a GMap function to draw the final boundaries of the polygons.

Properties of Generated MVGs: KMap ensures that each group is represented by a single polygon, containing only the nodes associated with that group. Since we create polygons based on the embedding of a group-group network, the proximity between polygons more accurately reflects the relationships among the groups. The independent layout of each subgraph captures the subgraph structures, but compromises the representation of between-group relationships. Since each subgraph fits into its square, the node distribution within each polygon may take an artificial square shape. The final polygons, created by partitioning the plane through GMap, may deviate from the proportional area property that was initially realized by the squares. However, each polygon should still approximate the area of its initial square as GMap places random points in the unbounded space when partitioning the plane to avoid large deviation (see Section 3.1). In extreme cases, the node distribution within each polygon will still provide an indication of its group size. Due to the independent layouts of subgraphs, the between-group edges may create a large number of edge crossings.

4.2 Method 2 — Polygon Constrained Layouts (PCL)

Method 2 takes the final polygons generated by KMap and distributes the nodes of the graph using a force-based layout with the constraint that they must remain inside their polygons and move according to the relationships within and outside of the polygons (Figure 5(right)). The challenge of drawing a graph within a polygon has been studied theoretically in the literature [22], though the focus has been on planar graphs without accounting for external influences beyond the polygon.

There are several challenges to tackle. (C_1) The nodes in the group must be distributed properly to fit and utilize the polygon area. (C_2) Each node should position itself based on both the neighbors within the polygon and outside the polygon without crossing the polygon boundary. (C_3) The polygon gives a constrained space for the nodes to move and hence the edges may turn out to be small. Therefore, we need force models that would ensure the edge lengths are not too small.

Our method will modify the force-based layout approach [16]. A typical force-based layout algorithm exerts a central gravity to keep the nodes close to the center of the display, defines a repulsion force between every pair of nodes, and an attraction force between adjacent vertices. To tackle the challenge (C_1), we modify the central gravitational force and create corner gravity that would stretch the group to fit its polygon. We also increase the repulsion gradually so that the convex hull of the group nodes covers at least 70% of the polygon area. For (C_2), we pull nodes towards the center of the external polygon, where the strength of the pull on a node is determined by its mass and the proportion of the external to internal neighbors. For (C_3), i.e., to ensure that the edges are sufficiently large, we control the displacement of the nodes based on a minimum edge length threshold. We now describe the details.

Gravitational Force. Let (x_C, y_C) be the center C of the layout. The gravitational force $\vec{F}_g$ acting on a node v with mass (degree) m_v positioned at (x_v, y_v) is typically defined as

$$\vec{F}_g = k_g \cdot m_v \cdot \frac{\vec{r}}{\|\vec{r}\|}, \tag{1}$$

where k_g represents a gravitational constant and $\vec{r} = (x_C - x_v, y_C - y_v)$ is the vector from v to the center C. We apply a gravitational force at the center of each polygon. The center is obtained by taking the average of the coordinates of the corners of the polygon, and the force acts on all the nodes that lie inside the polygon. Let v be a node inside a polygon P. Let $C(P)$ be the center of P and let O be the closest point of intersection between the ray $\overrightarrow{vC(P)}$ and the boundary of P. Then we modify the gravitational force as

$$\vec{F}_g = k_g \cdot m_v \cdot \frac{\vec{r}}{d(v,O)\vec{r}}, \tag{2}$$

where $d(v, O)$ is the Euclidean distance between v and O. This modification ensures that nodes that are closer to the boundary, i.e., with small $d(v, O)$, are pulled strongly towards the center, whereas the nodes that are closer to the center are not influenced much by gravity. In cases where a node is initially pushed outside the polygon due to repulsion, $d(v, O)$ remains small and gravity pulls the node back toward the polygon center. Figure 6(Left) and (middle) illustrates the difference between using and not using our gravity force, highlighting the impact of this force in shaping the visualization within the polygon.

Corner Gravity. We design the corner gravity so that the node distribution stretches further toward the polygon's corners. Let v be a node inside polygon P and let $c_1, \ldots, c_k$ be the corners of P. This force calculates a gravitational pull on v from each corner c_j, where $1 \le j \le k$, based on the distance $d(v, c_j)$. This corner gravity $\vec{F}_{g_j}$ exerted by a corner c_j is defined as

$$\vec{F}_{g_j} = k_g \cdot m_i \cdot \frac{\vec{r_j}}{\|\vec{r_j}\|}$$

where m_v is the mass of v and $\vec{r_j} = (x_{c_j} - x_v, y_{c_j} - y_v)$ represents the vector from v to c_j. The corner gravity is balanced with the center gravity and hence nodes do not clump together at a corner. Figure 6(Right) illustrates the combined effect of our central gravity and corner gravity, which gives a much better polygon coverage.

External Gravity. The external gravitational force is applied to the nodes that are connected to external polygons so that they are pulled toward the center of the external polygon. For a node v in a polygon P with a connection to the nodes of an external polygon Q with center q, the gravitational force $\vec{F}_q$ is designed by scaling a typical gravitational force by the ratio of the number of neighbors

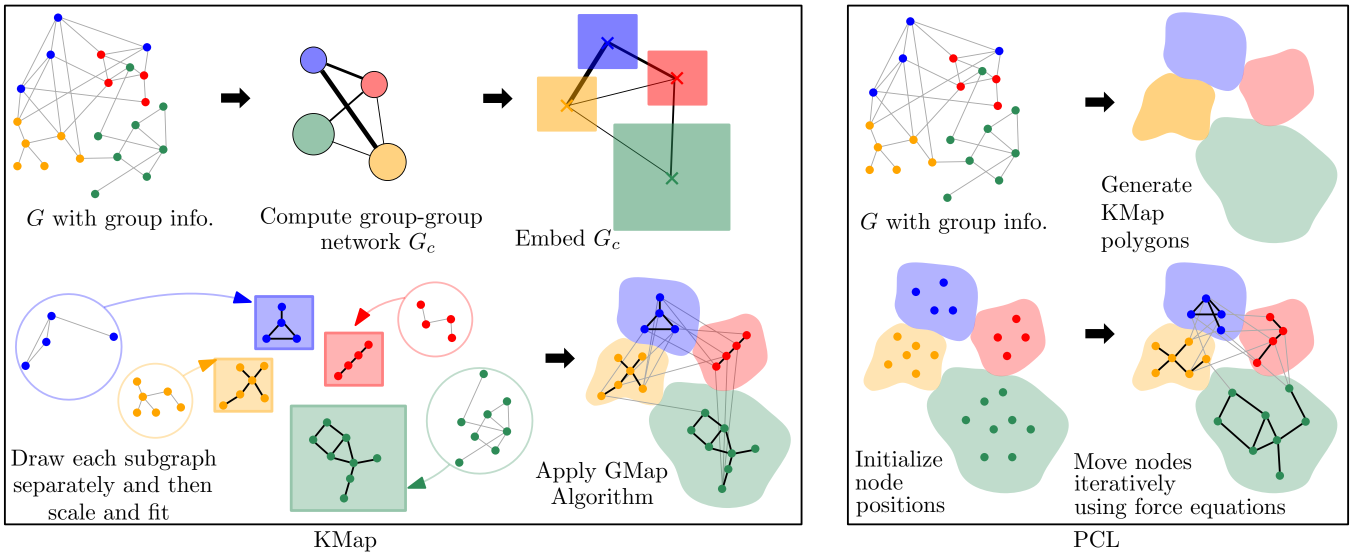

G with group info.
Compute group-group network G_c
Embed G_c
Draw each subgraph separately and then scale and fit
Apply GMap Algorithm
KMap
G with group info.
Generate KMap polygons
Initialize node positions
Move nodes iteratively using force equations
PCL

Figure 5: Conceptual representation for the steps of Method 1 (KMap) and Mathod 2 (PCL).

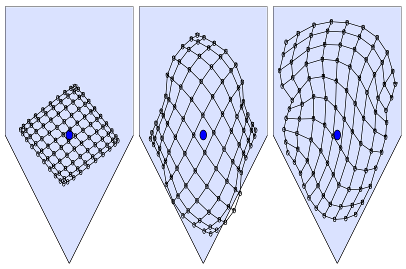


Figure 6: Left: A layout of a grid using the traditional force-based layout. Middle: A layout of the same grid with our customized gravitational force. Right: The effect of corner gravity.

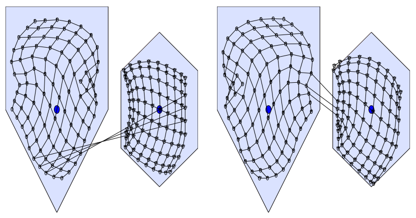


Figure 7: Left: A layout of two grids, each lying inside its own polygon. Without external gravity, we see longer edges that connect nodes between these polygons. Right: A layout of the same grids but with external gravity, which brings nodes of one polygon that are adjacent to the other polygon closer to the polygon boundary.

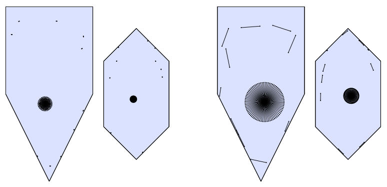


Figure 8: Left: A force layout with traditional attraction force on a graph containing a large star. Right: Application of the modified attraction force.

of v inside Q to that of P, i.e.,

$$\vec{F}_q = \frac{m'_v}{m_v} \cdot (k_g \cdot m_v \cdot \frac{\vec{r_e}}{\|\vec{r_e}\|}) = k_g \cdot m'_v \cdot \frac{\vec{r_e}}{\|\vec{r_e}\|}. \quad (3)$$

Here $\vec{r_e} = (x_v - x_q, y_v - y_q)$ is the vector from v to q, and m_v, m'_v are the number of neighbors of v inside P and q, respectively. Although from the equation it seems that all nodes with external connections will move towards the boundary of P, due to the presence of central gravity, nodes with larger m_v are less affected by the external gravity. This is a desired outcome as nodes with larger m_v are stronger representatives of P. Figure 7 illustrates the role of external gravity using two grids that reside in their polygons but are connected by a few edges.

Attraction Force. In a traditional force-based algorithm the larger the repulsion force, the larger is the layout. When the drawing area is unbounded, this naturally allows edges to stretch based on the force simulation. However, when the drawing area is limited by a polygonal region, we need to adjust the attraction force based on the space available. Let v and w be two adjacent nodes with $\Delta_x = |v_x - w_x|$ and $\Delta_y = |v_y - w_y|$. A typical attraction force is proportional to the distance between v and w, i.e., $\vec{F_a}(v, w) = (\Delta_x, \Delta_y)$. We modify the attraction force as follows. We first compute a minimum distance threshold α which is computed by dividing the total area of the polygon by the number of nodes. Intuitively, we want the neighbors of a node to spread at a distance of α. Then the new force is $\vec{F_a}(v,w) = (\Delta_x \frac{\alpha}{|vw|}, \Delta_y \frac{\alpha}{|vw|})$, where $|vw|$ is the distance between v and w. Figure 8 illustrates the result.

Properties of Generated MVGs: Since PCL utilizes the polygons generated by KMap, all polygon properties remain consistent with those of KMap. The nodes of a subgraph are initially positioned at the center of the polygon and moved gradually until they occupy a significant portion (α) of the polygon. Unlike KMap, this process enforces a soft constraint, which may occasionally result in nodes extending beyond the polygon boundaries. To prevent such occurrences, users can adjust α or increase the gravitational force constant. The node distribution within each polygon is influenced by relationships both within and outside the subgraphs, and hence between-group relationships are better captured. Therefore, PCL is likely to produce fewer edge crossings than KMap.

5 EXPERIMENTS

We conduct the experiments on real-life datasets. We first describe the datasets, and then discuss some examples of MVGs created by the six methods described in Section 3, and finally present a quantitative evaluation of these methods.

5.1 Dataset Description

We evaluate all our methods using real-world datasets and using quantitative metrics that measure the layout properties.

Small Graphs. This consists of the graphs musicland, bookland, university-similarity, tradeland and recipies from [21]. If a group is not specified in the graph information, then we create the groups by a community detection algorithm.

DBLP Graphs. For this dataset, we consider the time interval 2020–2023 and select 13 conference venues. We assign a person x to a venue y if the person has the highest number of publications in y than any other venues. We take 10 graphs by taking random samples of node size $|V| \in \{100, 300, 500, 700, 900\}$, and thus 50 graphs in total.

University Collaboration Graphs. The University of Arizona Knowledge Map (KMap) uses various types of research metadata, such as publications, proposals, grants, patents, and collaborative projects, to map the university's internal collaboration networks. In the KMap dataset, each node represents a researcher, and edges represent connections with other researchers on campus. This graph has 226 polygons. We create 10 random sample of this graph by taking $5i$ nodes per polygon, where $1 \le i \le 10$. If there is less than $5i$ nodes in a polygon, we take all its nodes.

5.2 Qualitative Assessment

Figure 9 illustrates a co-authorship network with 4787 nodes and 11489 edges. For large graphs, GMap and MapSet fail to create natural-looking country shapes. SF-CBA has better polygonal shapes but the polygons are in a sparse arrangement. Furthermore, the structure within individual groups is unclear. Figure 10 shows that KMap and PCL both produce readable layouts. KMap does not move the nodes based on between-group relations and hence we can see many small but dense subgraphs placed uniformly across each polygon. PCL captures the between-group relations better and the nodes move closer to the boundary due to their outside connections. This is also evident from the example shown in Figure 1 even for a graph that is 10 times larger. Figure 11 shows the MVGs created for the tradeland graph dataset. All methods produce readable layout for this small graph with 150 nodes and 134 edges, but group sizes are better captured by SF-CBA, KMap and PCL. The relations among groups are better captured by PCL.

5.3 Quantitative Results

Let $G = (V, E)$ be a graph with node set V and edge set E. Let $H = (\mathcal{V}, \mathcal{E})$ be a weighted graph, representing the group information, i.e., $\mathcal{V}$ corresponds to a partition of V into groups, and edge set $\mathcal{E}$ represents the relationships between two groups. We design some quality metrics (M1–M7) to capture the desired qualities of an MVG.

Metric M1 measures the convexity of the polygonal regions. User studies in prior work [17, 21] have demonstrated a preference for convexity, but no quantitative metric has been used to capture this quality.

Metrics M2–M6 are designed to capture the quality of the organization of the polygons. M2 assesses the extent to which polygonal regions cover the drawing area. This stems from the natural expectation of efficient area utilization. M3 examines how accurately the polygon adjacencies reflect the adjacencies in the group-group network. M4 measures how well the polygon areas reflect the weight of the groups. Both the contiguity of the regions and their area-proportionality are common constraints employed in cartographic representations [8, 9, 31], but they have not been quantitatively assessed in this specific context before.

Metrics M5–M7 are designed to capture the quality of a node-link layout. M5 (Stress) and M7 (Edge Crossing) are commonly used metrics in the literature. M5 measures the ratio of graph distance to display distance between pairwise nodes, while M7 counts the number of pairs of edges that create a crossing. M6 measures whether nodes are distributed in the polygon, which relates to readability of the layout and efficient utilization of the available drawing space.

In the following, we provide the details.

5.3.1 *Polygon Convexity (M_1).*

In an MVG, each node $v \in V$ is contained within a polygon $P \in \mathcal{V}$ that represents the group of v. We prefer the polygon to be similar to a convex polygon but with an irregular boundary. Let $c(P)$ be the number of corner-to-corner segments that are contained entirely in the polygon P. We measure the convexity of the polygon as follows.

$$M_1 = \frac{1}{|\mathcal{P}|} \sum_{P \in \mathcal{P}} (c(P)/|P|^2) \tag{4}$$

where $\mathcal{P}$ is the set of polygons representing $\mathcal{V}$. A larger value for M_1 reflects better polygon aesthetics.

5.3.2 *Drawing Area Coverage (M_2).*

We prefer the drawing area to be covered by the polygons. Hence we measure the convex hull of the union of the polygons to the sum of the area of the polygons.

$$M_2 = \frac{area(CH(\bigcup_{P \in \mathcal{P}}))}{\sum_{P \in \mathcal{P}} area(P)} \tag{5}$$

where $CH(X)$ is the convex hull of polygon P. A larger value for M_2 indicates better coverage.

5.3.3 *Polygon Neighborhood Preservation (M_3).*

The proximity of the polygons should be reflective of the graph H. In this metric, we

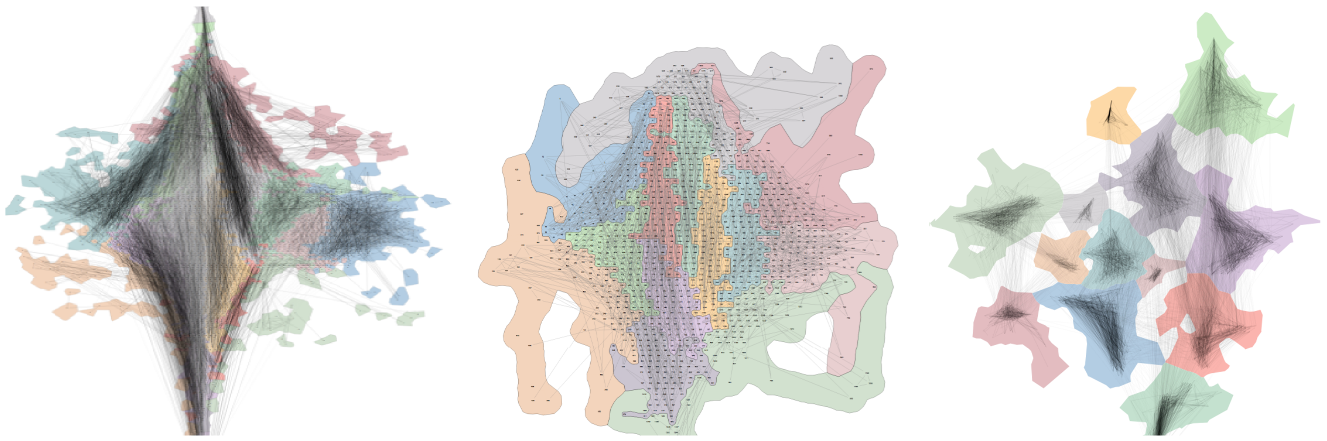


Figure 9: MVGs for a co-authorship network with about 11,000 edges: GMap, MapSet, and SF-CBA (left to right).

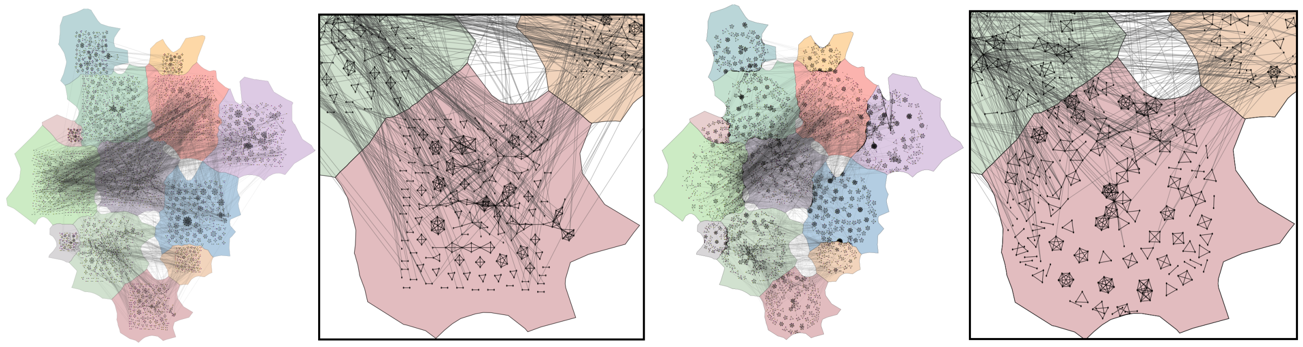


Figure 10: (left) KMap for the co-authorship network of Fig. 9 with a zoomed in view that shows uniform positions of the nodes. (right) PCL of the same network with a zoomed in view that repositions nodes to reveal structural information.

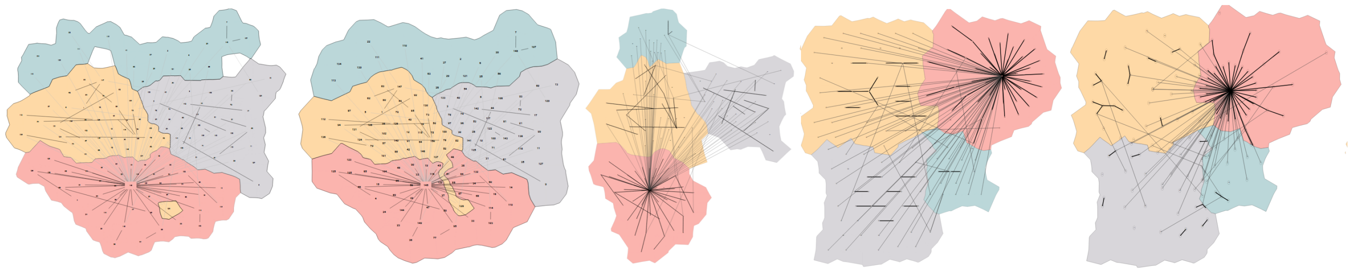


Figure 11: MVGs created (from left) by GMap, MapSet, SF-CBA, KMap, and PCL for the tradeland graph dataset.

strictly measure the adjacencies that have been correctly realized. Let $adj(u, v)$ be an indicator function that is 1 if the polygons of two vertices u and v of H are adjacent (i.e., touch or share part of their boundaries). Then the metric is as follows.

$$M_3 = \frac{1}{|E(H)|} \sum_{(u,v)\in E(H)} \mathbb{J}(u, v) \tag{6}$$

5.3.4 Polygon Area Realization (M_4). We prefer the area of each polygon to be proportional to the weight of the group (e.g., number of vertices) that it represents. We first measure the absolute difference $r(P)$ between the relative weight of a group and relative area of the corresponding polygon.

$$r(P) = \left|\frac{w(v)}{\sum_{v\in V(H)} w(v)} - \frac{area(P)}{\sum_{P\in\mathcal{P}} area(P)}\right|$$

where P is the polygon corresponding to v. We then subtract the mean area error from 1 to measure the polygon area realization.

$$M_4 = 1 - \frac{1}{|\mathcal{P}|} \sum_{P\in\mathcal{P}} r(P) \tag{7}$$

A larger value for M_4 indicates a lower area error.

5.3.5 Stress (M_5). The stress of a layout is a classic metric that compares the graph distance and display distance [27]. We use normalized stress, which has also been used in[21].

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

Here $m = \frac{1}{2} \sum_{uv} w(u,v)$ and $w(u,v)$ is set to d_{uv}. A layout is considered to be good if it has low stress. We set $M_5 = 1 - stress$ so that a larger value indicates higher quality.

5.3.6 Polygon Area Coverage (M_6). We want the polygon area to be utilized by the nodes inside the polygon. Hence we first find the error by subtracting the area of the convex hull of the vertices inside a polygon from the convex hull of the polygon. We then subtract the mean error from 1 to define M_6.

$$M_6 = 1 - \frac{1}{|\mathcal{P}|} \sum_{P \in \mathcal{P}} \frac{area(CH(P)) - area(f(P))}{area(CH(P))} \tag{8}$$

Here $f(P)$ is the area of the convex hull of the nodes in P.

5.3.7 Edge Crossing (M_7). This metric counts the number of pairs of edges that properly cross in the layout. We normalize the metric by dividing the number of crossings by the maximum possible crossings, i.e., the number of edges squared. We subtract the value from 1 so that a higher value indicates fewer crossings.

$$M_7 = 1 - \frac{1}{|E|^2} \sum_{e,e' \in E} I(e, e'), \tag{9}$$

where $I(e, e')$ is 1 if e and e' properly intersect, and 0 otherwise.

5.4 Results

Table 1 shows the performance of all methods for the Small Graphs dataset, where the metrics are designed such that larger values are better. Although GMap and MapSet do not produce the visualization that we prefer for an MVG, e.g., GMap contains fragmented polygons and MapSet contains polygons with narrow extensions, we also report their performances. They can be seen as extreme cases and by comparing them with the rest (SF-CBA, KMap, PCL) we can obtain an idea of how much the performances deviate when we incorporate group information explicitly when constructing a graph layout. GMap and MapSet perform very well in M2 (Drawing Area Coverage), M3 (Polygon Neighborhood Preservation), and M5 (Stress). While M2, M3 are likely due to their compact polygon packing, M5 is obvious as the whole graph layout already minimizes Stress. When comparing these metrics with the rest (SF-CBA, KMap, PCL), the performance degradation appears small (often within 10%).

When comparing SF-CBA, KMap, and PCL, we see KMap and PCL to win in M2 (Drawing Area Coverage), M3 (Polygon Neighborhood Preservation), M4 (Polygon Area Realization), M6 (Polygon Area Coverage), and M7 (Edge Crossing). This is expected for M2, M3, M4 and M6 as both KMap and PCL produce more compact layout than SF-CBA. KMap (and hence PCL) had high values in M1 (Polygon Convexity), which is consistent with our understanding as we have seen others to either have fragmented or irregular polygons. PCL had higher M7, i.e., lower edge crossing, than KMap which is due to the between-group edges that KMap does not carefully handle. Surprizingly, KMap had higher M5, i.e., lower stress, which we believe due to the graphs being small and not having too many between-group edges to produce noticeable differences and we will see the differences later when analyzing DBLP dataset. Table 2 shows the performances on the University Collaboration dataset. We see the same trends as in Table 1 except for M3 (Polygon Neighborhood Preservation) and M5 (Stress) when SF-CBA and PCL wins over KMap, respectively. Note that this dataset has over 200 polygons, which may generate favorable scenarios for M3 and M5 for those methods.

Figure 12 illustrates the performances on DBLP dataset. For many metrics, we see high variability in the performances. However, for M1 (Polygon Convexity), KMap (and hence PCL) are consistent winners, and for M7 (Edge Crossing), PCL appears to be consistently winning over all others. We performed a Wilcoxon paired test to determine whether there are significant differences between the means of various methods. For M1 (Polygon Convexity), we observed significant difference in all pairs. KMap (and hence PCL) had significantly higher mean than others ($p < 0.01$). For M7 (Edge Crossing), we observed PCL to have a significantly higher mean ($p < 0.01$) when compared with each other method. We did not find any statistical difference for any other metric except for M1 and M7. Although this is unfortunate, it shows the challenge of designing metrics that would properly capture the MVG properties.

5.5 Summary

For small graphs with a few hundred edges, GMap, MapSet, and SF-CBA are viable options for creating MVGs, as their outputs may be readable. However, GMap often produces fragmented polygons, MapSet tends to generate meandering polygon shapes, and SF-CBA creates complex polygonal structures similar to GMap. For large graphs with several thousand edges, our methods, KMap and PCL, are currently the only approaches capable of producing readable layouts. KMap is significantly faster than PCL, as it involves fewer force equations to simulate, whereas PCL requires more time to converge. Despite this, PCL excels at revealing between-group relationships with fewer edge crossings while preserving polygon convexity. Thus, PCL better achieves the desired properties of an MVG compared to other methods. However, in applications where visualizing between-polygon edges is not a priority, KMap offers a more organized visualization within individual polygons than PCL.

6 CONCLUSION

We examined the problem of creating map-like visualizations for graphs with group restrictions. The experimental results show that both the graph structure and between-group relationships are better captured in the visualizations created by our methods compared to the ones available in the literature. The source code of our implementation is available here [28]. Although the methods we examine produce layouts that visually look very different, the quantitative metrics are not much distinctive. Designing better quantitative metrics would be a natural direction for research.

We have deployed a system [19] to be used within the University of Arizona (Figure 3) that visualizes the faculty collaboration graph as an MVG where the polygons correspond to the departments.

	GMap		MapSet		SF-CBA		KMap		PCL	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
M1 (Polygon Convexity)	0.29	0.01	0.19	0.07	0.29	0.03	**0.30**	0.01	**0.30**	0.01
M2 (Drawing Area Coverage)	0.72	0.11	0.88	0.04	0.68	0.04	**0.75**	0.05	**0.75**	0.05
M3 (Polygon Neighborhood Preservation)	0.70	0.10	0.70	0.10	0.63	0.07	**0.65**	0.16	**0.65**	0.16
M4 (Polygon Area Realization)	0.92	0.02	0.93	0.02	0.94	0.02	**0.98**	0.00	**0.98**	0.00
M5 (Stress)	0.99	0.00	0.99	0.00	**0.99**	0.00	**0.99**	0.00	**0.99**	0.00
M6 (Polygon Area Coverage)	0.21	0.08	0.21	0.08	**0.22**	0.08	0.18	0.07	0.15	0.05
M7 (Edge Crossing)	0.92	0.08	0.92	0.08	0.92	0.09	**0.96**	0.07	**0.96**	0.07

Table 1: Performances for all metrics on the Small Graphs dataset. A larger value indicates better performance. The highest values in each row among the three methods (SF-CBA, KMap, PCL) are shown in bold.

	GMap		MapSet		SF-CBA		KMap		PCL	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
M1 (Polygon Convexity)	0.09	0.06	0.20	0.07	0.19	0.09	**0.33**	0.02	**0.33**	0.02
M2 (Drawing Area Coverage)	0.71	0.05	0.70	0.06	0.72	0.05	**0.73**	0.08	**0.73**	0.08
M3 (Polygon Neighborhood Preservation)	0.68	0.06	0.68	0.07	**0.72**	0.05	0.70	0.05	0.71	0.06
M4 (Polygon Area Realization)	0.92	0.01	0.93	0.02	0.91	0.01	**0.92**	0.01	**0.92**	0.01
M5 (Stress)	0.99	0.00	0.99	0.00	**0.99**	0.00	**0.99**	0.00	**0.99**	0.00
M6 (Polygon Area Coverage)	0.20	0.00	0.20	0.00	0.20	0.01	0.20	0.00	**0.21**	0.00
M7 (Edge Crossing)	0.92	0.04	0.91	0.04	0.91	0.05	0.92	0.02	**0.96**	0.02

Table 2: Performances for all metrics on University Collaboration dataset. A larger value indicates better performance. The highest values in each row among the three methods (SF-CBA, KMap, PCL) are shown in bold.

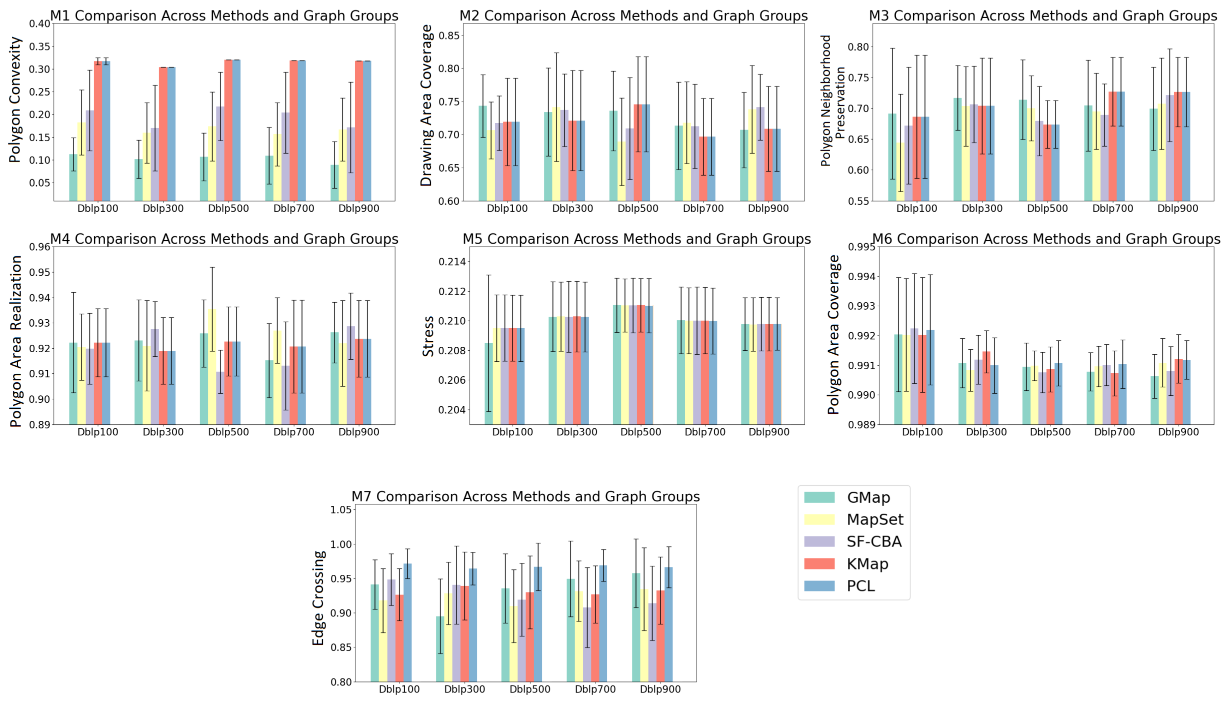


Figure 12: The mean and standard deviation of all metrics for all methods over the DBLP co-authorship graphs dataset.

We have had several meetings with such potential end-users to understand the pros and cons of using MVG. We have had several meetings with potential end-users to understand the pros and cons of using MVG. It appears that such an MVG can act as an institutional knowledge map, which is helpful to the administrators and research facilitators. Although MVG has its promise, there are also challenges associated with interpreting the map. When working with end-users, we often get some natural questions: (a) Why does a department appear at a central location of the map whereas some others appear in the periphery? (b) Why does department X have a smaller or larger area than department Y? (c) Does the relative position of the departments correspond to their physical location on campus? (d) Why are some polygons disconnected even when there are relations among the corresponding groups? Most of these questions are resolved when the users are explained the map metaphor for the graph, the properties of a layout, and the system's attempt to automatically create country-like regions for the departments under conflicting optimization goals. However, how this eventually influences the user perception and interpretation is yet to be explored. We envision full-fledged user studies on such real-life applications of MVG in subsequent papers.

We used SF-CBA, a coarse version of CBA [21], in our evaluation (due to the unavailability of a replication code for CBA). SF-CBA misses the part that brings the nodes towards their original position in the whole graph layout and moves the polygons closer together. One could attempt to revive CBA for a better evaluation. However, we believe due to the reliance of CBA on a whole graph layout, it would not be able to reveal individual group structure. It would be intriguing to create a variant of CBA that does not rely much on the whole graph layout but focus more on group information to construct better MVGs.

7 ACKNOWLEDGMENTS

The work of D. Mondal is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and that of S. Kobourov by the National Science Foundation (NSF CCF-2212130).

REFERENCES

[1] Basak Alper, Nathalie Riche, Gonzalo Ramos, and Mary Czerwinski. 2011. Design study of linesets, a novel set visualization technique. *IEEE transactions on visualization and computer graphics* 17, 12 (2011), 2259–2267.

[2] Kevin W Boyack, Richard Klavans, and Katy Börner. 2005. Mapping the backbone of science. *Scientometrics* 64, 3 (2005), 351–374.

[3] Stuart K Card, Jock Mackinlay, and Ben Shneiderman. 1999. *Readings in information visualization: using vision to think.* Morgan Kaufmann.

[4] Christopher Collins, Gerald Penn, and Sheelagh Carpendale. 2009. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE transactions on visualization and computer graphics* 15, 6 (2009), 1009–1016.

[5] DBLP. 2024. https://dblp.org/xml/release/. [Online; accessed 19-July-2024].

[6] Tim Dwyer, Kim Marriott, and Peter J Stuckey. 2006. Fast node overlap removal. In *Proceedings of the 13th International Symposium on Graph Drawing (GD).* Springer, 153–164.

[7] Alon Efrat, Yifan Hu, Stephen G. Kobourov, and Sergey Pupyrev. 2014. MapSets: Visualizing Embedded and Clustered Graphs. In *Graph Drawing - 22nd International Symposium, GD 2014, Würzburg, Germany, September 24-26, 2014, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 8871)*, Christian A. Duncan and Antonios Symvonis (Eds.). Springer, 452–463.

[8] Jared Albert Espenant and Debajyoti Mondal. 2024. StreamTable: An Area Proportional Visualization for Tables with Flowing Streams. *Computing in Geometry and Topology* 3, 1 (2024), 8–1.

[9] William Evans, Stefan Felsner, Michael Kaufmann, Stephen G Kobourov, Debajyoti Mondal, Rahnuma Islam Nishat, and Kevin Verbeek. 2018. Table cartogram. *Computational Geometry* 68 (2018), 174–185.

[10] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[11] Daniel Fried and Stephen G Kobourov. 2014. Maps of computer science. In *2014 IEEE Pacific Visualization Symposium.* IEEE, 113–120.

[12] Emden R Gansner, Yehuda Koren, and Stephen North. 2005. Graph drawing by stress majorization. In *Proceedings of the 12th International Symposium on Graph Drawing (GD).* Springer, 239–250.

[13] Marius Hogräfer, Magnus Heitzler, and Hans-Jörg Schulz. 2020. The state of the art in map-like visualization. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 647–674.

[14] Yifan Hu, Emden R Gansner, and Stephen Kobourov. 2010. Visualizing graphs and clusters as maps. *IEEE Computer Graphics and Applications* 30, 6 (2010), 54–66.

[15] Yifan Hu, Emden R. Gansner, and Stephen G. Kobourov. 2010. Visualizing Graphs and Clusters as Maps. *IEEE Computer Graphics and Applications* 30, 6 (2010), 54–66.

[16] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. 2014. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PloS one* 9, 6 (2014), e98679.

[17] Radu Jianu, Adrian Rusu, Yifan Hu, and Douglas Taggart. 2014. How to display group information on node-link diagrams: An evaluation. *IEEE Transactions on Visualization and Computer Graphics* 20, 11 (2014), 1530–1541.

[18] Tomihisa Kamada and Satoru Kawai. 1989. An algorithm for drawing general undirected graphs. *Information processing letters* 31, 1 (1989), 7–15.

[19] KMap. 2024. Inistituional Knowledge Map. https://kmap.arizona.edu/map. Accessed: 2024-11-12.

[20] Stephen G. Kobourov. 2007. *Force-Directed Drawing Algorithms.* Chapman & Hall/CRC.

[21] Stephen G. Kobourov, Sergey Pupyrev, and Paolo Simonetto. 2014. Visualizing Graphs as Maps with Contiguous Regions. In *16th Eurographics Conference on Visualization, EuroVis 2014 - Short Papers, Swansea, UK, June 9-13, 2014*, Niklas Elmqvist, Mario Hlawitschka, and Jessie Kennedy (Eds.). Eurographics Association.

[22] Anna Lubiw, Tillmann Miltzow, and Debajyoti Mondal. 2022. The complexity of drawing a graph in a polygonal region. *Journal of Graph Algorithms and Applications* 26, 4 (2022), 421–446.

[23] Wouter Meulemans, Nathalie Henry Riche, Bettina Speckmann, Basak Alper, and Tim Dwyer. 2013. Kelpfusion: A hybrid set visualization technique. *IEEE transactions on visualization and computer graphics* 19, 11 (2013), 1846–1858.

[24] Jacob Miller, Stephen Kobourov, and Vahan Huroyan. 2022. Browser-based hyperbolic visualization of graphs. In *2022 IEEE 15th Pacific Visualization Symposium (PacificVis).* IEEE, 71–80.

[25] Debajyoti Mondal, Manishankar Mondal, Chanchal K Roy, Kevin A Schneider, Yukun Li, and Shisong Wang. 2019. Clone-world: A visual analytic system for large scale software clones. *Visual Informatics* 3, 1 (2019), 18–26.

[26] Debajyoti Mondal and Lev Nachmanson. 2018. A New Approach to GraphMaps, a System Browsing Large Graphs as Interactive Maps. In *Proc. of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP-IVAPP).* 108–119.

[27] Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, Stephen G. Kobourov, and Jacob Miller. 2024. The Perception of Stress in Graph Drawings. In *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 320)*, Stefan Felsner and Karsten Klein (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 21:1–21:17. https://doi.org/10.4230/LIPIcs.GD.2024.21

[28] Ehsan Moradi. 2024. https://github.com/vga-usask/Map-Visualizations-For-Graphs.

[29] Ehsan Moradi and Debajyoti Mondal. 2023. BigGraphVis: Visualizing Communities in Big Graphs Leveraging GPU-Accelerated Streaming Algorithms. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, Christophe Hurter, Helen C. Purchase, and Kadi Bouatouch (Eds.). SCITEPRESS, 195–202.

[30] Lev Nachmanson, Roman Prutkin, Bongshin Lee, Nathalie Henry Riche, Alexander E Holroyd, and Xiaoji Chen. 2015. Graphmaps: Browsing large graphs as interactive maps. In *Proceedings of the 23rd International Symposium on Graph Drawing and Network Visualization (GD).* Springer, 3–15.

[31] Sabrina Nusrat, Md. Jawaherul Alam, and Stephen G. Kobourov. 2018. Evaluating Cartogram Effectiveness. *IEEE Trans. Vis. Comput. Graph.* 24, 2 (2018), 1077–1090. https://doi.org/10.1109/TVCG.2016.2642109

[32] Yosuke Onoue and Koji Koyamada. 2017. Optimal tree reordering for group-in-a-box graph layouts. In *SIGGRAPH Asia 2017 Symposium on Visualization.* 1–9.

[33] Alexandre Perrot and David Auber. 2020. Cornac: Tackling Huge Graph Visualization with Big Data Infrastructure. *IEEE Transactions on Big Data* 6, 1 (2020), 80–92.

[34] Peter Rodgers, Leishi Zhang, and Andrew Fish. 2008. General Euler diagram generation. In *Diagrammatic Representation and Inference: 5th International Conference, Diagrams 2008, Herrsching, Germany, September 19-21, 2008. Proceedings*

5. Springer, 13–27.

[35] Eduarda Mendes Rodrigues, Natasa Milic-Frayling, Marc Smith, Ben Shneiderman, and Derek Hansen. 2011. Group-in-a-box layout for multi-faceted analysis of communities. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. IEEE, 354–361.

[36] Paolo Simonetto, Daniel Archambault, David Auber, and Romain Bourqui. 2011. ImPrEd: An Improved Force-Directed Algorithm that Prevents Nodes from Crossing Edges. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1071–1080.

[37] André Skupin. 2002. A cartographic approach to visualizing conference abstracts. *IEEE Computer Graphics and Applications* 22, 1 (2002), 50–58.

[38] Steven van den Broek, Wouter Meulemans, and Bettina Speckmann. 2024. SimpleSets: Capturing Categorical Point Patterns with Simple Shapes. *IEEE Transactions on Visualization and Computer Graphics* (2024).

[39] Corinna Vehlow, Fabian Beck, and Daniel Weiskopf. 2017. Visualizing group structures in graphs: A survey. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 201–225.

[40] James A Wise, James J Thomas, Kelly Pennock, David Lantrip, Marc Pottier, Anne Schur, and Vern Crow. 1995. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. In *Proceedings of Visualization 1995 Conference*. IEEE, 51–58.

[41] Hsiang-Yun Wu, Martin Nöllenburg, and Ivan Viola. 2020. Multi-level area balancing of clustered graphs. *IEEE Transactions on Visualization and Computer Graphics* 28, 7 (2020), 2682–2696.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009