Layered Community Layout: Community Separation, Bridge Node Emphasis, and Clutter Reduction in Network Visualization

Jieqiong Zhao, Nabin Khanal, Sundas Qaiser, Cheryl Zhenyu Qian and Yingjie Victor Chen

Abstract—Large networks often become difficult to interpret because visual clutter obscures community structure and hides the roles that different nodes play within and across communities. We present *Layered Community Layout*, a network visualization design that reduces clutter and improves structural interpretability by explicitly separating inter-community and intra-community organization in a single overview. Our design represents a network as a set of communities identified through community detection and arranges them using a hierarchical display strategy. We structure each cluster with a three-layer concentric layout: nodes connected only within the cluster are placed in the interior, bridge nodes with both intra- and inter-community connections are moved to the periphery, and nodes connected only to bridge nodes are arranged around them. This layered structure reduces visual clutter while making node roles more visually explicit. Informed by Gestalt principles of proximity and similarity, the design supports rapid recognition of high-level community patterns as well as more targeted inspection of hubs, bridges, and outliers. Our approach shows how a role-specific, hierarchical layout can improve the readability of large, community structured networks while supporting both high-level overview and detailed analysis.

Index Terms—Network visualization, node-link diagrams, layout design.

1 Introduction

Many real-world networks exhibit a community-based organization in which nodes form densely connected groups linked by comparatively sparse cross-community ties. This structure arises in a wide range of domains, including research collaboration networks, where laboratories operate as relatively distinct yet interacting groups, and organizational networks, where departments such as engineering, marketing, and product management function as separate but coordinated units. Within these communities, nodes often assume different structural roles: some occupy central positions that sustain local cohesion, some appear as leaf nodes with only limited local attachment, and others act as bridges that connect otherwise separated groups. These complementary roles shape coordination, information flow, and vulnerability in the network, making them important for understanding both local organization and system-level integration.

Despite their analytical importance, community structure and node roles are often difficult to perceive in visualizations of large networks. As network size and density increase, visual clutter can obscure both the boundaries between communities and the roles that individual nodes play within and across them. Analysts must therefore contend with a tension between seeing the network as a collection of cohesive local groups and understanding it as an integrated global system. This challenge motivates visualization approaches that support the simultaneous perception of community boundaries, cross-community relationships, and node roles in a single view.

Current computational approaches to community detection seek to partition networks into groups that are more densely connected internally than externally, thereby exposing mesoscale structure that would otherwise be difficult to interpret in large graphs. Widely used methods include modularity-based approaches such as Newman's formulation [35] and the Louvain algorithm, [5] later improved by Leiden [42] to produce improved connected and more reliable partitions, as well as flow based methods [40]. Once communities have been identified, however, visualizing large community-structured networks remains challenging because conventional node-link diagrams and force-directed layouts quickly become cluttered as graph size and density increase [16]. Node-link diagrams become difficult to read in dense networks because of heavy edge crossings and occlusion; earlier studies suggest that their effectiveness declines when the number of links reaches about three to four times the number of nodes [33]. Prior work has explored several strategies for improving the readability of community structure, including cluster emphasized energy models such as LinLog [36], hybrid representations such as NodeTrix for dense subnetworks [22], and edge bundling methods that expose high-level connectivity while reducing visual clutter [25].

Multiscale systems such as GMine [27] and Graph Mapping [26] further demonstrate the value of hierarchical community aggregation and drill-down interaction for navigating large graphs. However, these approaches primarily support scalable exploration of clustered structure rather than explicitly encoding the structural roles of nodes within communities in a single overview. As a result, analysts can navigate between levels of abstraction, yet still struggle to distinguish which nodes function as community core nodes, which appear as leaf nodes, and which serve as bridges across community boundaries. To address this gap, we present *Layered Community Layout*, a role-specific network visualization design that visually separates inter-community and intra-community organization in a unified overview. Our design arranges each community using a layered concentric structure in which community core nodes occupy the interior, bridge nodes are moved to the periphery, and leaf nodes are placed in an outer layer according to their structural attachment within the community. By making these node roles visually explicit while preserving community context, the design reduces clutter and supports both high-level community analysis and detailed inspection of role-specific structure.

Our contributions are as follows:

- A role-specific community visualization design that separates inter-community and intra-community structure in a single overview.
- A multi-stage layered layout method that places bridge nodes, community core nodes, and leaf nodes according to their structural roles to improve community separation, reduce clutter, and improve role legibility.
- Case studies showing how the design supports the interpretation of community structure and node roles in large networks.

2 Related Work

2.1 Network Clustering Algorithms

Large networks are often easier to interpret when their intermediate scale structure is made explicit, especially when nodes are organized

• *Jieqoing Zhao is with Augusta University. E-mail: jiezhao@augusta.edu.*
• *Nabin Khanal, Sundas Qaiser, Cheryl Zhenyu Qian and Yingjie Victor Chen are with Purdue University. E-mail: {khanaln, sqaisor, qianz, victorchen}@purdue.edu.*
• *Correspondences to Yingjie Victor Chen. E-mail: victorchen@purdue.edu.*

Manuscript received xx xxx. 202x; accepted xx xxx. 202x. Date of Publication xx xxx. 202x; date of current version xx xxx. 202x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.202x.xxxxxxx

into communities that reveal how densely connected groups relate to one another. Community detection, therefore, provides a structural scaffold for visualization: detected communities become the main visual components, guide inter-community separation, and support the interpretation of node roles within and across communities.

Widely used methods include modularity based approaches, such as Newman's formulation [35], Louvain [5], and Leiden [42], as well as flow based methods [40]. Because these methods differ in how they define and extract communities, no single approach is universally optimal across all network types [15].

For visualization, community detection is important because it converts a large network into an intermediate scale structure that supports grouping, comparison, and abstraction. In particular, detected communities can guide layout organization, separate dense regions, and reveal how nodes contribute to both within- and between-community structure. In our work, community detection serves as a preprocessing step for the proposed layered community layout, allowing the visualization to move beyond a flat node-link diagram by organizing the network around communities and the roles nodes play within them.

2.2 Network Visualization Methods

Network visualization comprises a rich design space of representations for relational data, in which nodes encode entities and edges encode connections. Prior work has shown that this design space extends well beyond conventional node-link diagrams and includes adjacency matrices, arc diagrams, scatterplot-based layouts, geographic network views, small multiples, and hybrid techniques such as NodeTrix [22]. These representations provide complementary views of network structure and support different analytical goals, such as tracing paths, inspecting dense connectivity patterns, comparing groups, or relating topology to spatial and temporal context [8, 14, 41].

Alongside representational choices, network visualization also depends on layout and ordering strategies. Existing methods include force-directed layouts [16, 20], geometric layouts [28], matrix seriations [3], and other procedural arrangements that position nodes according to topology, attributes, or ordering constraints [41]. Because these methods emphasize different aspects of structure, such as local connectivity, dense subgraphs, or global organization, no single visualization approach is appropriate for all network types or tasks [14].

As graph size and density increase, however, conventional node-link diagrams become difficult to read because overlap, edge crossings, and occlusion quickly obscure network structure, a phenomenon often referred to as the *hairball* problem [33, 49]. To address this limitation, researchers have proposed abstraction and aggregation strategies that simplify network structure at multiple levels. In hierarchical visualization methods, clusters or groups can be treated as higher-level visual components, allowing communities to be abstracted as aggregate nodes and displayed through nested or grouped structures such as trees [2], circles [44], or bundled hierarchies [24].

Taken together, this broader design space shows that layout is not merely a technical step for placing nodes, but a central design decision that shapes what structural patterns become visible. Compared with these approaches, our work focuses on community structured networks and contributes a role-specific layout that uses communities as the main organizing components while making node roles within and across communities more explicit.

2.3 Visualization Methods for Networks with Communities

Once communities have been identified, visualization methods can use them as intermediate scale components for organizing layout, separating dense regions, and supporting comparison across groups. Community structured visualization extends beyond showing individual nodes and edges by making group structure an explicit part of the representation. This is particularly useful for large networks, where community structure can provide an additional level of abstraction [11, 12].

Prior work has proposed several strategies for visualizing networks with communities. Some methods adapt layout algorithms so that communities appear as visually distinct regions [37]. Other methods make group structure explicit through additional visual encodings layered on top of an existing layout. Bubble Sets, for instance, use isocontours to reveal set relations and visually enclose related nodes without changing their underlying positions [10]. More broadly, grouped and clustered graph visualizations have represented communities as explicit higher level structures, such as maps [17], enclosing regions [44, 48], or multiple scale abstractions [30], to support comparison and overview.

Compared with prior community structured visualization methods, our work focuses more explicitly on structural roles within and between communities. Existing approaches reveal clusters and reduce clutter, but they less directly distinguish nodes that support internal cohesion from nodes that mediate cross community connectivity. Our layered layout addresses this gap by organizing bridge, core, and leaf nodes as separate visual roles within a customized overview.

3 Visualization Design

To make the design intuitive for analysts and consistent with the structural properties of the network, we first define the concepts of communities and node roles, then introduce the visual design that reflects community structured layout. Figure 1 summarizes the visual encodings, and Table 1 outlines the corresponding design strategies, tasks, and benefits.

3.1 Community and Node Roles

A *community* is a subset of nodes whose internal connectivity is denser, stronger, or more frequent than its connectivity to the remainder of the network. Communities therefore constitute relatively cohesive mesoscale substructures within a larger graph and are commonly interpreted as groups of nodes that interact preferentially with one another rather than with outsiders [29, 35].

A *bridge node* is a node that connects two or more communities and thereby occupies a boundary spanning position in the network [7, 18]. Such nodes are important because they support inter-community connectivity and facilitate intra-community communication, resource exchange, and influence diffusion. In contrast to core nodes, bridge nodes are often only moderately embedded within any single community, yet they are disproportionately important for global connectivity because they link otherwise weakly connected or disconnected groups. As a result, bridge nodes often play a brokerage role: they mediate interactions across structural gaps and may serve as critical pathways for information flow between communities [7, 18]. Their removal can therefore weaken or even sever inter-community connections, making them potential points of structural vulnerability.

A *core node* is a node that is strongly embedded within a community, typically with many intra-community connections and comparatively few external links [6, 39]. Core nodes form the central structural backbone of a community. They tend to occupy central positions within the local cluster, are highly connected to other members of the same group, and are frequently surrounded by neighbors that are themselves mutually well connected. In this sense, core nodes exhibit high within-community embeddedness and high local structural importance, because they help maintain the internal cohesion and stability of the community. At the same time, they generally have limited connectivity between communities and therefore play only a minor brokerage role relative to bridge nodes.

A *leaf node* is a peripheral node with very low degree, often connected to the rest of the community through only one edge or a very small number of edges. In graph theoretic terms, the simplest case is a single degree vertex, also called a *pendant* or *end* vertex [45, 46]. In community-based network analysis, leaf nodes lie on the outer boundary of the community and contribute little to its internal cohesion. They typically do not add much redundancy to local connectivity and do not serve as important conduits for inter-community exchange. Because they often depend on a single neighboring node for attachment to the graph, removing that connection isolates them immediately. Consequently, leaf nodes can be interpreted as marginal, weakly integrated, or newly attached members of the community rather than as central or brokerage elements.

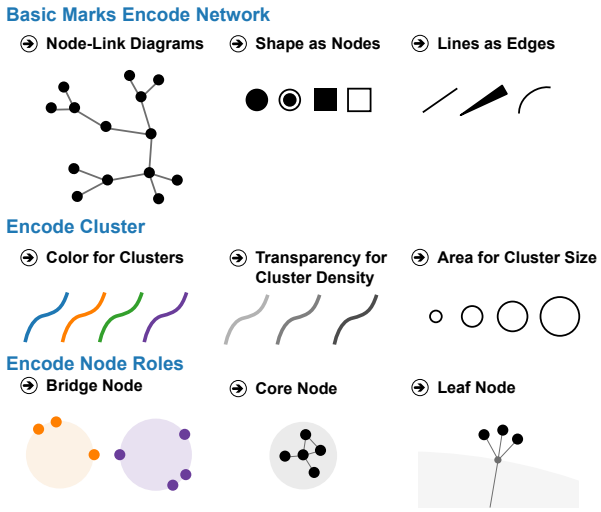

Fig. 1: Visual encodings for representing network structure, community-level attributes, and role-specific node representations.

3.2 Data Processing

To prepare network data for the proposed community-based visualization, we employ a multi-stage data processing pipeline that transforms the raw graph into a role-specific community structure. The goal of this pipeline is to partition the network into communities and then classify nodes within each community according to their structural roles, namely *bridge nodes*, *leaf nodes*, and *core nodes*. Because the visualization is designed to emphasize both community structure and boundary spanning connectivity, we adopt an *overlapping community detection* strategy rather than a strictly disjoint partitioning approach. Overlapping community detection is appropriate in this context because nodes in real-world networks may participate in multiple groups simultaneously, especially when they serve as interfaces between otherwise distinct communities [1, 19, 38].

Let the input network be an undirected graph

$$G = (V, E),$$

where V is the set of nodes and E is the set of edges. We first detect a set of overlapping communities

$$\mathscr{C} = \{C_1, C_2, \ldots, C_k\},$$

where each $C_i \subseteq V$. Because membership may overlap, a node can belong to more than one community, which is useful for preserving nodes that naturally serve as shared connectors between groups [1, 38].

For each node $v \in C_i$, we compute its intra-community degree

$$d_{\text{in}}(v, C_i) = |\{u \in C_i \mid (v,u) \in E\}|,$$

and its inter-community degree

$$d_{\text{out}}(v, C_i) = |\{u \in V \setminus C_i \mid (v,u) \in E\}|.$$

These two measures are used for node-role classification.

A node is classified as a *bridge node* if it has at least X connections to nodes outside its community:

$$d_{\text{out}}(v, C_i) \geq X.$$

Here, X is a user-defined threshold controlling the sensitivity of bridge detection. A node is classified as a *leaf node* if it has exactly one intra-community connection:

$$d_{\text{in}}(v, C_i) = 1.$$

Table 1: Overview of the design strategies and analytical benefits.

High-Level Task	Goal	Design Strategy	Benefit
Scalability [23, 50]	Large networks	Abstract overview with selected bridges	Simplified but informative summary
Clutter reduction [13, 33]	Dense graphs	Hierarchical layout and layered placement	Improved readability
Multi-level analysis [22, 31]	Overview and detail	Community glyphs with role-specific placement	Supports broad and targeted analysis
Community separation [22, 37]	Community structure	Circular glyphs, community color, global layout	Clear group boundaries
Role distinction [6]	Core / bridge / leaf	Interior / boundary / exterior placement	role-specific interpretation
Cross-community links [7]	Bridge structure	Perimeter bridge placement	Explicit brokerage structure
Internal organization [16, 28]	Within-community structure	Compact core placement	Reveal cohesive circles in cluster
Size comparison [4, 9]	Community scale	Glyph area for node size, transparency for edge density	Magnitude comparison
Special cases	Chains and nodes shared across communities	Dedicated handling	Better structural fidelity

All remaining nodes are classified as *core nodes*. Thus, a node is labeled as a core node if

$$d_{\text{out}}(v, C_i) < X \quad \text{and} \quad d_{\text{in}}(v, C_i) > 1.$$

In summary, the preprocessing pipeline consists of three steps: detecting overlapping communities, computing intra- and inter-community degree for each node, and assigning node roles according to the above rules. This process produces a role-specific community structure that directly supports the proposed visualization by preserving both the modular organization of the network and the functional differences among nodes.

3.3 Overall Design

We propose a community-focused representation for network visualization in which each community is encoded as a single circular glyph. This design abstracts the internal composition of a community while preserving important structural distinctions among different node roles, allowing viewers to interpret both intra-community organization and inter-community connectivity at a glance.

In this representation, each community is depicted as a colored circular glyph. The use of a circular enclosure serves two purposes. First, it provides a clear visual boundary that distinguishes one community from another. Second, it offers a structured spatial frame within which different categories of nodes can be arranged according to their functional roles in the network. Color is used as a primary identifier for communities, enabling rapid perceptual separation among groups and supporting comparisons across the network.

Within each circular community glyph, nodes are arranged according to a role-specific spatial hierarchy. Bridge nodes are positioned along the circumference of the circle. This placement reflects their boundary spanning function, as these nodes are responsible for connecting one community to nodes in other communities. Locating bridge nodes on the perimeter makes this role visually explicit and facilitates the drawing and tracing of inter-community links. Core nodes are placed inside the circle, occupying the interior region of the glyph. This

interior placement communicates their centrality to the community's internal structure, emphasizing that these nodes primarily participate in within-community relationships. In contrast, leaf nodes are positioned outside the circle. Their external placement indicates their peripheral status, as leaf nodes typically have limited connectivity and are often attached to the community through only one or a few relationships. This inside-boundary-outside arrangement establishes a consistent and semantically meaningful mapping between topological role and spatial location.

The overall size of each circular glyph represents the total number of nodes contained in the corresponding community. More specifically, the area of the circular node is designed to scale with community size, so that larger communities appear visually larger than smaller ones. Encoding community size by area provides an intuitive quantitative cue and allows users to estimate the relative scale of communities without inspecting exact counts. This choice supports overview tasks such as identifying dominant communities, comparing group magnitudes, and recognizing imbalances in the network structure.

A key aspect of the design lies in how the radius of the community circle is determined. In principle, the circle size should reflect the number of nodes that define the community's internal substance, particularly the core nodes. However, the geometric requirements of the layout introduce an important constraint: the circle must also be sufficiently large to accommodate bridge nodes distributed along its boundary. Because bridge nodes are arranged around the circumference, a community with many bridge nodes requires a larger perimeter to avoid overlap and maintain legibility. Consequently, the radius cannot be determined solely by the number of core nodes. Instead, it is computed through a balancing process that accounts for the relative quantities of leaf nodes, bridge nodes, and core nodes.

This balancing process reflects the dual objective of the design: faithful quantitative encoding and effective spatial organization. On the one hand, we seek to make the community area, and therefore its square root of radius, correspond as closely as possible to the amount of internal content represented by the core structure. On the other hand, we must ensure that the boundary has enough capacity to place bridge nodes clearly and evenly. Thus, the final radius emerges from a compromise between semantic proportionality and layout feasibility. In cases where the number of bridge nodes is high, the radius may be larger than what would be implied by the number of core nodes alone. This adjustment is necessary to preserve readability, prevent boundary congestion, and maintain a stable visual grammar across communities of varying structural compositions.

The placement of leaf nodes outside the circular boundary further reinforces this compositional logic. Since leaf nodes are peripheral and do not contribute to inter-community bridging, positioning them outside the glyph avoids cluttering the interior and preserves the circle as a meaningful container for the community's core and boundary structure. At the same time, their proximity to the circle maintains their visual association with the community to which they belong. This strategy creates a layered representation such that the interior signifies the community core, the boundary signifies external interfaces, and the exterior signifies peripheral attachments.

Overall, the proposed design provides a compact yet expressive way to visualize community-based networks. By aggregating each community into a structured circular glyph, the representation reduces visual complexity while retaining critical information about community size, internal composition, and external connectivity. The design is especially suitable for networks in which understanding the roles of nodes within and across communities is more important than displaying every node in a conventional force-directed layout. Through its role-specific spatial organization and size encoding, the visualization supports both high-level overview and detailed structural interpretation

3.4 Abstract Design

In addition to the role-specific community glyph, we introduce an *abstract community glyph* for scalable overview of large community-based networks. This design summarizes each community as two visual components: an *internal aggregate*, which encodes the community's internal structure, and an *external interface*, formed by explicitly displayed bridge nodes that preserve the dominant inter-community topology. Rather than rendering all internal nodes individually, the glyph represents community size through area and internal edge density through color intensity, enabling compact comparison of community scale and cohesion. Bridge nodes remain visible because they define the principal pathways between communities and therefore preserve the global backbone of the network.

Compared with the detailed role-specific glyph, this abstraction deliberately sacrifices fine-grained information about individual core and leaf nodes in exchange for improved readability and scalability. It is particularly suited to overview and exploratory tasks in which analysts need to identify major communities, compare their internal connectivity, and trace the main bridging structure across the network. In this way, the abstract community glyph complements the detailed view by emphasizing structural summarization rather than internal role composition.

3.5 Special Cases

3.5.1 Long Chains

A long chain is a linear network topology in which nodes are connected sequentially through single edges. Long chains are visually and structurally special because they encode linear rather than clustered organization. Whereas communities emphasize cohesion and density, chains emphasize sequence, fragility, and dependency. This makes them important patterns to preserve or highlight in community-based network visualization, as their significance may be lost if they are abstracted in the same manner as compact groups.

3.5.2 One Common Node Connecting Multiple Communities

In complex networks, some nodes maintain substantial connections to multiple communities and thus function as bridges between them. To model this structure, we can employ overlapping community detection methods, such as clique percolation [38], mixed-membership stochastic blockmodels [1], and overlapping label propagation [19], all of which allow nodes to belong to more than one community. These methods are particularly appropriate for capturing boundary spanning nodes whose structural roles are distributed across communities.

In our visualization, such a node is shown in every community to which it belongs. Instead of a standard circular mark, it is represented as a pie-chart glyph whose slices encode community membership. Each slice is sized according to the node's relative connectivity to the corresponding community, measured as the proportion of its links to nodes in that community. Thus, if a node belongs to three communities and has connection weights a, b, and c, its membership proportions are $a/(a+b+c)$, $b/(a+b+c)$, and $c/(a+b+c)$, and the angular span of each slice is determined accordingly.

3.6 Interaction

To support analysis and sensemaking in large community structured networks, our two-level visualization incorporates interaction techniques that help analysts navigate across scales while managing visual complexity. We provide *multi-level expand and collapse* operations for communities, allowing analysts to abstract a community into a single aggregate node while preserving its connections to other communities, and to progressively reintroduce detail through either the layered community view or the original node-link representation [27, 34]. This interaction supports movement between overview and detail without losing structural context. We also support *selective edge visibility*, enabling analysts to independently show or hide intra-community and inter-community edges in order to reduce clutter and foreground specific relational patterns [21, 47]. Finally, *zooming and panning* allow users to examine the network at multiple levels of scale, from the global arrangement of communities to detailed local structures. Collectively, these interactions enhance large-network sensemaking by reducing

clutter, preserving context, and enabling focused examination of both community-level organization and node-level roles.

4 Layout Algorithm

We compute the visualization layout using a hierarchical force-directed algorithm that operates on a reduced community graph while simultaneously optimizing the angular positions of bridge nodes on each community boundary, as illustrated in Figure 2. The method is designed to satisfy two objectives: (1) to produce a clear global arrangement of communities based on inter-community connectivity, and (2) to place bridge nodes along the boundary of each community so that they indicate connection directions in a stable and interpretable manner.

4.1 Community-Level Abstraction

Let the original network be partitioned into a set of communities. Each community is treated as a single composite node, yielding a higher-level graph in which vertices correspond to communities rather than individual nodes. Edges in this reduced graph are induced by inter-community links, that is, by connections between bridge nodes belonging to different communities. This abstraction reduces the layout problem from the full node-link network to a more compact representation that preserves the backbone of community-level structure.

Each community node is represented as a circle whose position is defined by its center in the plane and whose radius is determined by the visual encoding described earlier. Bridge nodes are constrained to lie on the circumference of the corresponding community circle and are parameterized by angular coordinates. Therefore, the layout optimization involves two coupled sets of variables: the Cartesian positions of community centers and the angular positions of bridge nodes on community boundaries.

4.1.1 Global Community Placement

At the global level, we position communities using a force-directed layout on the reduced community graph. Repulsive forces are applied between all pairs of community nodes to prevent overlap and maintain spatial separation, while attractive forces are applied along inter-community connections induced by bridge nodes. We weight these attractive forces by the number of bridge-node links between communities, such that strongly connected communities are placed closer together. This yields a layout in which spatial proximity reflects inter-community connectivity while preserving overall readability.

However, weighting attraction solely by the number of inter-community links can introduce an undesirable bias when communities of very different sizes connect to the same neighboring community. In such cases, a smaller community often has fewer links to the shared neighbor than a larger community does, and therefore experiences a weaker attractive force. As a result, the smaller community may be pushed disproportionately far away, even when it should remain visually associated with the same region of the network. From a perceptual and layout standpoint, this is undesirable: smaller communities can usually be placed closer to their neighbors without harming readability, whereas larger communities require more separation because they occupy more visual space.

To address this issue, we modulate the repulsive force according to the sizes of the two community nodes. Specifically, larger communities exert stronger mutual repulsion than smaller ones, allowing compact communities to remain closer to related neighbors while preserving sufficient spacing among larger aggregates. We model this size dependent repulsion using an inverse-square formulation:

$$Fr_{ij} = a \cdot \frac{S_i S_j}{r_{ij}^2},$$

where S_i and S_j denote the sizes of communities C_i and C_j, r_{ij} is the distance between them, and a is a tunable scaling constant.

Similarly, we incorporate community size into the computation of attraction forces. The attraction force is defined as

$$Fa_{ij} = b \cdot (S_i S_j) \cdot N_c \cdot r_{ij}^2,$$

Data: undirected graph $G = (V, E)$, bridge threshold X

/***role-specific data processing***/
detect communities $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$;
foreach *community* $C_i \in \mathscr{C}$ **do**
  foreach *node* $v \in C_i$ **do**
    $d_{\text{in}}(v, C_i) \leftarrow |\{u \in C_i \mid (v,u) \in E\}|$;
    $d_{\text{out}}(v, C_i) \leftarrow |\{u \in V \setminus C_i \mid (v,u) \in E\}|$;
    if $d_{\text{out}}(v, C_i) \geq X$ **then**
      assign v the label *bridge node* in C_i;
    else
      if $d_{\text{in}}(v, C_i) = 1$ **then**
        assign v the label *leaf node* in C_i;
      else
        assign v the label *core node* in C_i;
      end
    end
  end
end
/***layout generation***/
/*step 1: community-level abstraction*/
initialize the reduced community graph $G_c = (\mathscr{C}, E_c)$;
foreach *pair of communities* (C_i, C_j) **do**
  if C_i *and* C_j *share at least one bridge node link* **then**
    add edge (C_i, C_j) to E_c;
    $w_{ij} \leftarrow$ number of bridge node links between C_i and C_j;
  end
end
foreach *community* $C_i \in \mathscr{C}$ **do**
  initialize the center position p_i;
  determine the radius r_i;
  initialize each bridge node in C_i an angular coordinate θ;
end
/*coupled iterative optimization*/
while *not converged* **do**
  /*community update*/
  foreach *pair of communities* (C_i, C_j) **do**
    apply a repulsive force between C_i and C_j;
  end
  foreach *edge* $(C_i, C_j) \in E_c$ **do**
    apply an attractive force;
  end
  update community center positions $\{p_i\}$;
  /*bridge node update*/
  foreach *community* $C_i \in \mathscr{C}$ **do**
    foreach *bridge node* $b \in C_i$ **do**
      update the angle θ of b on the boundary of C_i;
    end
    enforce angular spacing among bridge nodes in C_i;
  end
end
/*step 2: intra-community node placement*/
foreach *community* $C_i \in \mathscr{C}$ **do**
  fix p_i, r_i, and bridge nodes positions;
  /*core nodes placement*/
  initialize core nodes near the center position p_i;
  while *the core node layout in* C_i *has not converged* **do**
    apply repulsive forces among core nodes;
    apply attraction on core-core and core-bridge edges;
    update core nodes positions and keep them inside C_i;
  end
  /*leaf nodes placement*/
  foreach *leaf node* $l \in C_i$ **do**
    find the unique adjacent node u of l;
    place l on outward radial line from center p_i through u;
    apply a local spacing adjustment if needed;
  end
end

Fig. 2: The proposed Layered Community Layout algorithm.

where N_c is the number of connections between the two communities, and b is a scaling constant.

When communities share common nodes (Section 3.5.2), we introduce an additional stronger attraction between the corresponding node slices. This force is stronger than that induced by ordinary bridge-node links, ensuring that overlapping communities remain spatially close in the final layout. The resulting optimization produces a community-level arrangement that preserves the topology of the reduced graph, highlights overlap, and maintains visual clarity.

4.1.2 Boundary-Constrained Bridge Node Optimization

While community centers are being positioned, bridge nodes are optimized along the circumference of their respective community circles. Unlike ordinary node-link layouts in which node positions vary freely in two-dimensional space, each bridge node in our method is restricted to move only along a circular boundary. This constraint preserves the semantic interpretation of bridge nodes as boundary elements mediating communication between communities.

For a bridge node, the preferred angular position is determined by the direction from its host community to the external community to which it connects. Intuitively, a bridge node should occupy the point on the circle that faces the neighboring community most directly. Therefore, during optimization, each bridge node rotates along the boundary toward the angular position aligned with its connection target. This reduces visual ambiguity by making the direction of inter-community connectivity immediately apparent.

When a bridge node connects to multiple external nodes or communities, its preferred angle may be computed from the weighted average of the corresponding directions. In this way, the final angular position reflects the dominant orientation of its external relationships rather than any single edge alone.

4.1.3 Spacing Constraints on the Boundary

Because multiple bridge nodes may belong to the same community, an additional spacing mechanism is required to prevent them from collapsing onto the same location or becoming visually indistinguishable. We therefore introduce a local repulsion or angular separation constraint among bridge nodes on the same circumference. This constraint preserves a minimum distance between adjacent bridge nodes in angular space, ensuring that each occupies a unique and legible position.

The boundary optimization can thus be understood as balancing two competing factors. On the one hand, each bridge node is pulled toward the angular direction of its external connection target. On the other hand, neighboring bridge nodes repel each other along the circle to maintain spacing. The equilibrium of these two forces produces a stable boundary arrangement that is both informative and readable.

An important consequence of this formulation is that bridge nodes connecting to the same neighboring community naturally move toward nearby angular positions. Because they share similar target directions, they tend to form localized groups along the circumference of the host community. These groups provide an additional visual cue: they reveal which region of a community boundary is associated with a given neighboring community and make bundles of related inter-community links easier to interpret.

4.1.4 Coupled Iterative Optimization

The layout process is iterative because community placement and bridge-node positioning are mutually dependent. Changes in community positions alter the directions toward neighboring communities, which in turn change the preferred angular locations of bridge nodes. Conversely, the arrangement of bridge nodes affects how inter-community links are visually expressed and therefore contributes to the perceived quality of the global layout.

The algorithm alternates between these two levels of optimization:

1. **Community update:** compute repulsive and attractive forces among community nodes and update their positions in the plane.
2. **Bridge-node update:** for each community, update the angular positions of bridge nodes on the circumference according to the directions of their connected target communities, while enforcing boundary spacing constraints.
3. **Repeat** until the movement of community centers and bridge-node angles falls below a specified threshold or a maximum number of iterations is reached.

This alternating optimization allows the global and local structures to co-evolve toward a consistent final configuration. Communities settle into positions determined by their connectivity patterns, while bridge nodes align themselves around each boundary in ways that reflect those same patterns.

4.1.5 Algorithmic Properties

The final layout has several desirable properties. First, it preserves the integrity of the community as a visual unit by positioning each community as a whole rather than distributing its members independently throughout the display. Second, it makes the structure of inter-community relationships legible by placing connected communities near one another. Third, it encodes connection direction explicitly by positioning bridge nodes on the side of the community facing the communities they connect to. Finally, it reveals groups of bridge nodes associated with the same neighboring community through spatial proximity along the boundary.

Compared with a standard force-directed layout applied directly to all nodes, this method provides a more structured and scalable solution for community-based network visualization. The reduction of the network to a community graph simplifies the global layout problem, while the boundary-constrained optimization of bridge nodes preserves important information about how communities interact. As a result, the algorithm supports both high-level structural overview and detailed interpretation of inter-community connectivity.

4.2 Intra-Community Node Placement

After the community-level layout converges and the bridge nodes have been positioned along the circumference of each community, we perform a second-stage layout to place the remaining nodes within each community. This stage focuses on the internal organization of a single community while keeping the global arrangement fixed. In particular, the positions of community centers, community radii, and bridge nodes are treated as fixed constraints and are no longer updated. The purpose of this stage is to reveal the internal structure of each community by first arranging the core nodes in the interior of the community and then attaching the leaf nodes around the exterior.

4.2.1 Placement of Core Nodes

Core nodes are placed within the interior region of the community using a constrained force-directed procedure. At this stage, the bridge nodes serve as anchored boundary conditions that influence the positions of the core nodes but do not themselves move. The layout of the core nodes is therefore driven by two types of relationships: (1) edges between core nodes and bridge nodes, and (2) edges among core nodes themselves.

To initialize the layout, core nodes are assigned provisional positions near the center of the community circle. This initialization reflects their semantic role as the internal structural backbone of the community and provides a stable starting point for subsequent optimization. A force-directed model is then applied within the boundary of the community. Core nodes repel one another locally to avoid overlap and to maintain sufficient spacing for visual readability. At the same time, attractive forces are introduced along edges, drawing connected pairs closer together. These attractive forces operate both among core nodes and between core nodes and the already-fixed bridge nodes on the circumference.

Because the bridge nodes remain fixed, they act as positional anchors that pull connected core nodes toward the corresponding regions of the community interior. As a result, the placement of a core node reflects not only its relationships to other core nodes but also its role in supporting specific boundary connections. Core nodes that connect strongly to a particular set of bridge nodes tend to shift toward the side

of the interior facing those bridge nodes, while still remaining inside the community circle. Similarly, groups of densely interconnected core nodes tend to remain close to one another, forming coherent internal substructures. The resulting layout balances internal cohesion with boundary-oriented structure.

The optimization is subject to a geometric constraint that all core nodes must remain inside the community circle. If an update would move a core node outside the permitted interior region, its position is projected back into the valid region or otherwise corrected by the constraint mechanism. In this way, the layout preserves the interpretation of the community as a bounded container whose interior is occupied by the core structure and whose boundary is occupied by bridge nodes.

4.2.2 Placement of Leaf Nodes

Once the positions of the core nodes have stabilized, we place the leaf nodes. Unlike the core nodes, leaf nodes do not participate in the internal force-directed optimization because they typically have only a single connection and do not contribute substantially to the internal structural organization of the community. Instead, each leaf node is positioned directly according to its unique adjacent node.

Each leaf node position is computed from the position of the node to which it is connected, which may be either a bridge node or a core node. Specifically, the leaf node is placed along the outward radial direction defined by its adjacent node relative to the community center. This construction ensures that the leaf node remains visually associated with the community while clearly indicating its peripheral status. Leaf nodes connected to bridge nodes are placed outside the community boundary, whereas leaf nodes connected to core nodes are placed near those core nodes within the community boundary. If multiple leaf nodes are attached to the same anchor node, small angular offsets or local spacing adjustments may be introduced to avoid overlap and to preserve the visibility of individual leaf nodes.

This placement strategy gives leaf nodes a clear and lightweight visual role. Because they are positioned after the internal structure has already been established, they do not distort the layout of the core or bridge nodes. At the same time, their locations communicate their dependency on the nodes to which they attach and reinforce the layered organization of the community: bridge nodes on the boundary, core nodes in the interior, and leaf nodes adjacent to bridge nodes placed outside the community boundary.

4.2.3 Two-Stage Intra-Community Layout

The intra-community layout can therefore be summarized as a two-stage procedure:

1. Fix the positions of the bridge nodes on the community circumference and place the core nodes inside the community using a constrained force-directed layout driven by core-to-core and core-to-bridge edges.
2. After the core node layout converges, place each leaf node according to the position of its unique adjacent node. Leaf nodes connected to bridge nodes are placed outside the community boundary, whereas leaf nodes connected to core nodes are placed near those core nodes within the community boundary.

This staged design improves both stability and interpretability. By anchoring bridge nodes before the internal layout begins, the algorithm preserves the directional structure established during the community-level layout. By placing core nodes before leaf nodes, it ensures that the most structurally important internal relationships determine the organization of the community interior. Finally, by computing leaf node positions only after the interior has stabilized, the method preserves the readability of the community glyph while maintaining role-specific peripheral placement.

4.3 Resulting Structure

The final configuration yields a layered intra-community representation. Bridge nodes remain fixed on the circumference, where they indicate the directions of inter-community connectivity. Core nodes occupy the interior, where their positions reflect both their mutual relationships and their connections to the bridge nodes. Leaf nodes connected to bridge nodes are placed outside the community boundary. Together, these constraints produce a coherent and interpretable layout in which the internal structure of each community is organized around the previously established boundary geometry.

5 Case Studies

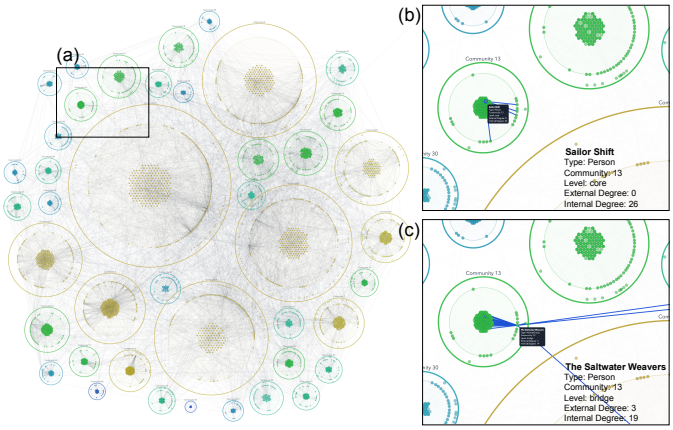

Fig. 3: Illustration of the 2025 VAST Challenge MC1 music industry dataset. (a) Our layered community layout reveals Sailor Shift's structural role within Community 13. (b) She appears as a highly connected core node embedded in a dense local community, while the surrounding layout makes it easier to distinguish direct collaborators, other influential entities, and the broader community context for tracing potential indirect influence. (c) Highlights the node *Musical Group*, which has many internal links and three links to other communities.

5.1 Music Industry in 2025 VAST Challenge MC1 Dataset

We applied our layout to the 2025 VAST Challenge MC1 dataset [43], a person centric knowledge graph of the music industry containing artists, artworks (e.g., albums and singles), collaborators, and influence relations such as stylistic and lyrical influence. The dataset contains 5,000 nodes and 14,708 edges. Using the Louvain community detection algorithm, we identified 76 communities, with community sizes ranging from 1 to 403 nodes.

A central narrative in this dataset is the success and rising influence of *Sailor Shift*. One of the key analytical questions in the challenge asks: *"Who has she collaborated with, and whom has she directly or indirectly influenced?"* Our layout supports this question by first locating Sailor Shift within the broader community structure. Through search and visual inspection, we find that she belongs to Community 13 (Figure 3(a)). Once highlighted in the layout, Sailor Shift appears as a highly connected core node with 26 internal links and no external links (Figure 3(b)). This immediately reveals an important structural insight: her collaborations are concentrated within her own community rather than spread across multiple communities. In other words, her prominence in the dataset is expressed not through brokerage across communities, but through strong embeddedness and influence within a single tightly connected group.

This interpretation is much harder to obtain from a conventional node-link layout, where dense local connectivity can easily dissolve into clutter. In contrast, our layered design makes Sailor Shift's role visually explicit. Because core nodes are positioned in the interior of a community, her placement near the center, together with her dense internal connections, directly signals her importance to the cohesion of Community 13. The layout therefore helps distinguish a structurally central artist from a bridge node that would instead connect multiple communities from the boundary.

We also show how the layout supports the search for indirect influence. After identifying Sailor Shift's direct collaborators, the analyst can expand outward from her community and inspect neighboring structures. A quick glance at the overview reveals another node with exceptionally high connectivity within the same region. The hover

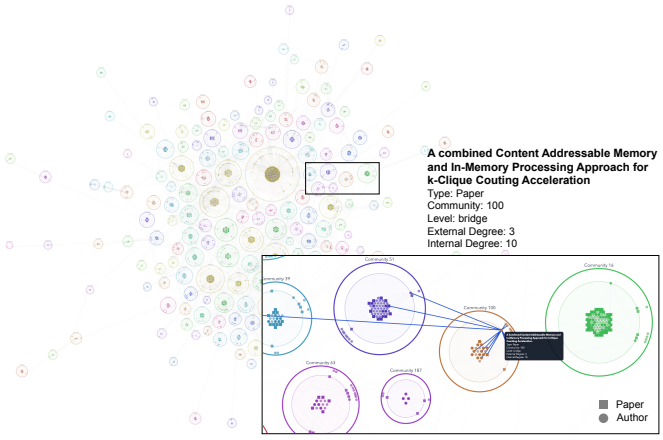

Fig. 4: Illustration of the DBLP co-authorship network. The proposed layout preserves readability in a large collaboration graph with 8,267 nodes, 138,080 links, and 197 communities, enabling analysts to distinguish tightly connected research groups, central collaborators within communities, and bridge researchers across communities.

interaction identifies this node as a *Musical Group* connected to many artists (Figure 3(c)). Because our layout separates core, bridge, and peripheral nodes, such highly connected entities immediately stand out as structurally important nodes. This allows analysts to move from a specific question about one artist to a broader understanding of the surrounding influence network, identifying which entities likely mediate collaboration, amplify influence, or anchor the local music ecosystem.

5.2 Citation Network in the DBLP Dataset

We further evaluate our layout using a citation network derived from the DBLP dataset [32]. Our processed graph contains 8,267 nodes, 13,808 links, and 197 communities. Papers form the citation backbone, while authors provide additional structural context. The DBLP network reflects scholarly influence through citation structure, where dense regions often correspond to coherent research areas and cross-community links indicate exchange across subfields.

To derive the community structure, we first partition the network based on paper-to-paper citation links and then assign non-paper nodes to communities according to the dominant community memberships of their neighboring papers. We subsequently classify nodes into bridge, leaf, and core roles. Nodes with at least two inter-community links are treated as bridge nodes, nodes with exactly one intra-community link are treated as leaf nodes, and all remaining nodes are treated as core nodes. This dataset highlights a central analytical task in bibliometric network analysis: distinguishing entities that are structurally central within a research area from those that connect otherwise separated areas. Our layout supports this distinction directly. At the global level, the community structured organization provides a clear overview of the major research clusters and their connectivity patterns. Within each community, papers or authors with strong local influence tend to occupy core positions in the interior, reflecting their dense within-community connections. By contrast, entities that connect multiple communities are placed along the boundary as bridge nodes, making cross-community links visually explicit. This spatial separation enables analysts to distinguish local prominence from cross-community brokerage without extensive edge tracing in a dense global network.

The DBLP network also demonstrates the utility of the layout for multi-level analysis. Starting from a paper or author of interest, analysts can first determine whether the node is embedded within a single research community or positioned between multiple communities. The analyst can then examine adjacent bridge nodes to identify likely pathways of topical transfer, interdisciplinary exchange, or citation flow across research areas. Because the layout preserves community boundaries while emphasizing bridge structure, it supports both focused inspection of individual entities and broader reasoning about the global organization of scholarly influence.

6 Discussion and Limitation

6.1 Complexity Analysis

We compare the complexity of a conventional force-directed layout on the full graph with that of our hierarchical two-level strategy.

Let the original graph contain n nodes and m edges. Suppose the graph is partitioned into k communities. For community i, let b_i, c_i, and ℓ_i denote the numbers of bridge, core, and leaf nodes, respectively, and let

$$n_i = b_i + c_i + \ell_i, \qquad \sum_{i=1}^{k} n_i = n.$$

We also denote by m^{inter} the number of inter-community edges, and by m_i^{cc} and m_i^{cb} the numbers of core–core and core–bridge edges inside community i.

6.1.1 Baseline: Full Force-Directed Layout

A standard force-directed layout on the full graph requires repulsive force computation over all node pairs and attractive force computation over all edges. Therefore, one iteration costs

$$O(n^2 + m).$$

If T_g iterations are required, the total cost is

$$O\Big(T_g(n^2 + m)\Big).$$

For sparse graphs, this is typically dominated by the quadratic term $O(T_g n^2)$.

6.1.2 Our Two-Level Approach

Our method separates the layout into a community-level stage and an intra-community stage.

Level 1: Community-Level Layout. At the first level, each community is treated as a single composite node. The reduced graph therefore contains k nodes instead of n. A force-directed layout on this reduced graph requires:

- repulsion among community nodes: $O(k^2)$,
- attraction induced by inter-community links: $O(m^{\text{inter}})$,
- angular adjustment of bridge nodes on community boundaries: $O\big(\sum_{i=1}^{k} b_i^2\big)$ in the naive implementation, due to spacing constraints among bridge nodes within the same community.

Hence, one iteration of the first stage costs

$$O\left(k^2 + m^{\text{inter}} + \sum_{i=1}^{k} b_i^2\right).$$

If this stage runs for T_1 iterations, its total cost is

$$O\left(T_1\left(k^2 + m^{\text{inter}} + \sum_{i=1}^{k} b_i^2\right)\right).$$

Level 2: Intra-Community Layout. After the community layout converges, bridge nodes are fixed and each community is processed independently.

For community i, only the core nodes participate in the force-directed optimization. One iteration costs

$$O(c_i^2 + m_i^{cc} + m_i^{cb}),$$

where $O(c_i^2)$ accounts for repulsion among core nodes, and the edge terms account for attractive forces along core–core and core–bridge edges. If $T_{2,i}$ iterations are required for community i, the total cost is

$$O\Big(T_{2,i}(c_i^2 + m_i^{cc} + m_i^{cb})\Big).$$

Summing over all communities gives

$$O\left(\sum_{i=1}^{k} T_{2,i}(c_i^2 + m_i^{cc} + m_i^{cb})\right).$$

Leaf nodes are then placed directly from the positions of their adjacent bridge or core nodes. Since each leaf node is positioned once, this step costs

$$O\left(\sum_{i=1}^{k} \ell_i\right) = O(n).$$

Therefore, the total complexity of the second stage is

$$O\left(\sum_{i=1}^{k} T_{2,i}(c_i^2 + m_i^{cc} + m_i^{cb}) + n\right).$$

Overall Complexity. Combining the two stages, the total complexity of our method is

$$O\left(T_1\left(k^2 + m^{\text{inter}} + \sum_{i=1}^{k} b_i^2\right) + \sum_{i=1}^{k} T_{2,i}(c_i^2 + m_i^{cc} + m_i^{cb}) + n\right).$$

6.1.3 Comparison

The conventional force-directed layout has complexity

$$O\Big(T_g(n^2 + m)\Big),$$

where the dominant term arises from repulsion over all nodes. In contrast, our method replaces the single global quadratic term n^2 with smaller quadratic terms defined over communities and their internal node sets:

$$k^2, \qquad \sum_{i=1}^{k} b_i^2, \qquad \sum_{i=1}^{k} c_i^2.$$

This decomposition is advantageous when the graph exhibits clear community structure, since these sums are typically much smaller than n^2.

In particular, if communities are relatively balanced, then

$$\sum_{i=1}^{k} c_i^2 \ll n^2 \quad \text{and} \quad \sum_{i=1}^{k} b_i^2 \ll n^2,$$

so the hierarchical method is substantially more efficient than a direct all-node force-directed layout. The improvement comes from three factors: reducing the global layout to k community nodes, confining quadratic repulsion to local community subproblems, and placing leaf nodes in linear time without iterative optimization.

The asymptotic gain is most significant for networks with strong modular structure. In the worst case, if nearly all nodes belong to one community, the complexity approaches that of the full layout. However, for the community-based networks targeted by our design, the hierarchical formulation provides a more scalable alternative in both theory and practice.

6.2 Design Tradeoffs

A central tradeoff of our design is that placing bridge nodes on the community boundary makes their cross-community role more visible, but can also introduce additional edge crossings near cluster boundaries. Rather than minimizing all crossings uniformly, our layout prioritizes structural interpretability by aligning node placement with node role. Despite this tradeoff, the layout reduces clutter overall through its hierarchical strategy. It first stabilizes the global arrangement of communities, and then organizes cohesive local structure within each cluster. This separation of global and local layout improves readability by preserving clear community boundaries and compact internal organization.

6.3 Limitations and Future Work

One limitation arises when a community contains many bridge nodes. Because these nodes are placed along the boundary, the area of a community must be enlarged to provide sufficient perimeter space and reduce overlap. In such cases, community area may reflect not only community node size, but also the number of bridge nodes that must be accommodated. Another limitation occurs in networks with many inter-community links. Even when bridge nodes are separated onto the periphery, dense external connectivity can still produce congestion around cluster boundaries. In addition, the effectiveness of the layout depends on the quality of the community detection result, since communities provide the structural scaffold for the visualization.

Several directions could further improve the proposed layout. One is to develop adaptive strategies for bridge node placement in communities with high boundary density, so that the layout can better balance role visibility and edge crossings. Another is to integrate edge bundling methods to reduce congestion among inter-community links while preserving the distinction between local and global structure. It would also be valuable to explore more flexible cluster sizing rules that better balance internal node count and boundary requirements. Moreover, future work could examine how alternative definitions of node roles and different community detection methods affect the readability and analytical value of the resulting layout across different network domains.

7 Conclusion

We presented *Layered Community Layout*, a network visualization design for large community structured graphs that reduces visual clutter by explicitly separating inter-community and intra-community organization. By combining a hierarchical layout strategy with role-specific node placement, the design makes community structure, bridge nodes, and leaf nodes more visually interpretable in a single overview. In particular, the layout uses communities as the main organizing components, places core nodes in compact interior regions, repositions bridge nodes to the periphery, and arranges leaf nodes around them to reflect their structural roles.

Through the design space, algorithmic implementation, and case studies, we showed that the role-specific layout can support both a high-level overview and a detailed structural interpretation. The proposed design helps analysts identify cohesive communities, distinguish node roles, and interpret cross-community connectivity more effectively than a flat node-link layout. Our case studies further demonstrate that the approach can scale to real-world networks while preserving interpretable structure across different application domains. More broadly, our work shows that readability in large networks depends not only on reducing clutter, but also on making meaningful structural distinctions visually explicit. By organizing the layout around communities and node roles, *Layered Community Layout* offers a structured alternative to conventional network layouts for large, complex graphs.

References

[1] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. In *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 21 of *NIPS 2008*, 8 pages. Curran Associates, Red Hook, NY, 2008. 3, 4

[2] D. Archambault, T. Munzner, and D. Auber. GrouseFlocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, Aug. 2008. doi: 10.1109/TVCG.2008.34 2

[3] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. *Computer Graphics Forum*, 35(3):693–716, July 2016. doi: 10.1111/cgf.12935 2

[4] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, Madison, WI, Nov. 2010. 3

[5] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. doi: 10.1088/1742-5468/2008/10/P10008 1, 2

[6] S. P. Borgatti and M. G. Everett. Models of core/periphery structures. *Social Networks*, 21(4):375–395, Oct. 2000. doi: 10.1016/S0378-8733(99)00019-2 2, 3

[7] R. S. Burt. Structural holes and good ideas. *American Journal of Sociology*, 110(2):349–399, Sept. 2004. doi: 10.1086/421787 2, 3

[8] I.-X. Chen and C.-Z. Yang. *Visualization of Social Networks*, pp. 585–610. Springer, New York, Oct. 2010. doi: 10.1007/978-1-4419-7142-5_27 2

[9] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984. doi: 10.1080/01621459.1984.10478080 3

[10] C. Collins, G. Penn, and S. Carpendale. Bubble Sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009. doi: 10.1109/TVCG.2009.122 2

[11] J. D. Cruz, C. Bothorel, and F. Poulet. Community detection and visualization in social networks: Integrating structural and semantic information. *ACM Transactions on Intelligent Systems and Technology*, 5(1), art. no. 11, 26 pages, Jan. 2014. doi: 10.1145/2542182.2542193 2

[12] S. Datta and E. Adar. CommunityDiff: Visualizing community clustering algorithms. *ACM Transactions on Knowledge Discovery from Data*, 12(1), art. no. 11, 34 pages, Feb. 2018. doi: 10.1145/3047009 2

[13] C. Dunne, S. I. Ross, B. Shneiderman, and M. Martino. Readability metric feedback for aiding node-link visualization designers. *IBM Journal of Research and Development*, 59(2/3):14:1–14:16, 2015. doi: 10.1147/JRD.2015.2411412 3

[14] V. Filipov, A. Arleo, and S. Miksch. Are we there yet? a roadmap of network visualization from surveys to task taxonomies. *Computer Graphics Forum*, 42(6), art. no. e14794, 31 pages, Sept. 2023. doi: 10.1111/cgf.14794 2

[15] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016. doi: 10.1016/j.physrep.2016.09.002 2

[16] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, Nov. 1991. doi: 10.1002/spe.4380211102 1, 2, 3

[17] E. R. Gansner, Y. Hu, and S. Kobourov. GMap: Visualizing graphs and clusters as maps. In *Proceedings of the IEEE Pacific Visualization Symposium*, PacificVis 2010, pp. 201–208, Mar. 2010. doi: 10.1109/PACIFICVIS.2010.5429590 2

[18] M. S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, May 1973. doi: 10.1086/225469 2

[19] S. Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10), art. no. 103018, 26 pages, Oct. 2010. doi: 10.1088/1367-2630/12/10/103018 3, 4

[20] L. Han, B. Wang, and S. Wang. Force-directed graph layout based on community discovery and clustering optimization. In R. Su, Y. Zhang, H. Liu, and A. F Frangi, eds., *Medical Imaging and Computer-Aided Diagnosis*, pp. 561–571. Springer, Singapore, Dec. 2023. 2

[21] J. Heer and D. Boyd. Vizster: Visualizing online social networks. In *Proceedings of the IEEE Symposium on Information Visualization*, InfoVis 2005, pp. 32–39. IEEE, Los Alamitos, Oct. 2005. doi: 10.1109/INFVIS.2005.1532126 4

[22] N. Henry, J.-D. Fekete, and M. J. McGuffin. Nodetrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007. doi: 10.1109/TVCG.2007.70582 1, 2, 3

[23] I. Herman, G. Melancon, and M. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, Jan. 2000. doi: 10.1109/2945.841119 3

[24] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, Oct. 2006. doi: 10.1109/TVCG.2006.147 2

[25] D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):983–990, 2009. doi: 10.1111/j.1467-8659.2009.01450.x 1

[26] D. Jonker, S. Langevin, D. Giesbrecht, M. Crouch, and N. Kronenfeld. Graph mapping: Multi-scale community visualization of massive graph data. *Information Visualization*, 16(3):190–204, Aug. 2017. doi: 10.1177/1473871616661195 1

[27] J. F. R. Jr., H. Tong, A. J. M. Traina, C. Faloutsos, and J. Leskovec. GMine: A system for scalable, interactive graph visualization and mining. In *Proceedings of the International Conference on Very Large Data Bases*, VLDB '06, pp. 1195–1198. ACM, New York, Sept. 2006. https://dl.acm.org/doi/10.5555/1182635.1164242. 1, 4

[28] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, Apr. 1989. doi: 10.1016/0020-0190(89)90102-6 2, 3

[29] F. R. Khawaja, Z. Zhang, Y. Memon, and A. Ullah. Exploring community detection methods and their diverse applications in complex networks: A comprehensive review. *Social Network Analysis and Mining*, 14(1):115, June 2024. doi: 10.1007/s13278-024-01274-1 2

[30] S. Langevin, D. Jonker, D. Giesbrecht, and M. Crouch. Multi-scale community visualization of massive graph data. *Proceedings of the exploring graphs at scale (EGAS)*, 4 pages, 2015. 2

[31] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, 5 pages. ACM, New York, May 2006. doi: 10.1145/1168149.1168168 3

[32] M. Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In A. H. F. Laender and A. L. Oliveira, eds., *String Processing and Information Retrieval*, 10 pages. Springer, Berlin, Heidelberg, Sept. 2002. doi: 10.1007/3-540-45735-6_1 8

[33] G. Melancon. Just how dense are dense graphs in the real world? a methodological note. In *Proceedings of the AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, 7 pages. ACM, New York, May 2006. doi: 10.1145/1168149.1168167 1, 2, 3

[34] L. Nachmanson, R. Prutkin, B. Lee, N. H. Riche, A. E. Holroyd, and X. Chen. Graphmaps: Browsing large graphs as interactive maps. In *Graph Drawing and Network Visualization*, GD 2015, pp. 3–15. Springer, Cham, Nov. 2015. doi: 10.1007/978-3-319-27261-0_1 4

[35] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, Apr. 2006. doi: 10.1073/pnas.0601602103 1, 2

[36] A. Noack. Energy models for graph clustering. In *Proceedings of the International Symposium on Graph Drawing*, vol. 4372 of *LNCS*, pp. 425–436. Springer, 2007. doi: 10.1007/978-3-540-70904-6_41 1

[37] A. Noack. Modularity clustering is force-directed layout. *Physical Review E*, 79(2), art. no. 026102, Feb. 2009. doi: 10.1103/PhysRevE.79.026102 2, 3

[38] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, June 2005. doi: 10.1038/nature03607 3, 4

[39] P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha. Core-periphery structure in networks (revisited). *SIAM Review*, 59(3):619–646, Jan. 2017. doi: 10.1137/17M1130046 2

[40] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, Jan. 2008. doi: 10.1073/pnas.0706851105 1, 2

[41] J. Scott-Brown, A. Pister, and B. Bach. NetPanorama: A declarative grammar for network construction, transformation, and visualization. *arXiv preprint arXiv:2310.18902v2*, 12 pages, Feb. 2025. doi: 10.48550/arXiv.2310.18902 2

[42] V. A. Traag, L. Waltman, and N. J. van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019. doi: 10.1038/s41598-019-41695-z 1, 2

[43] VAST Challenge Organizers. Vast challenge 2025 mini-challenge 1, 2025. https://vast-challenge.github.io/2025/MC1.html Accessed: 2026-03-30. 7

[44] C. Vehlow, F. Beck, and D. Weiskopf. Visualizing group structures in graphs: A survey. *Computer Graphics Forum*, 36(6):201–225, 2017. doi: 10.1111/cgf.12872 2

[45] E. W. Weisstein. Pendant vertex. Wolfram MathWorld, 2025. https://mathworld.wolfram.com/PendantVertex.html (Accessed: 2026-03-30). 2

[46] D. B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, 5 ed., May 2010. 2

[47] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. In *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No.03TH8714)*, pp. 51–58, Oct. 2003. doi: 10.1109/INFVIS.2003.1249008 4

[48] Y. Wu, W. Wu, S. Yang, Y. Yan, and H. Qu. Interactive visual summary of major communities in a large network. In *Proceedings of the IEEE Pacific Visualization Symposium*, PacificVis 2015, pp. 47–54, Apr. 2015. doi: 10.1109/PACIFICVIS.2015.7156355 2

[49] V. Yoghourdjian, D. Archambault, S. Diehl, T. Dwyer, K. Klein, H. C. Purchase et al. Exploring the limits of complexity: A survey of empirical studies on graph visualisation. *Visual Informatics*, 2(4):264–282, Dec. 2018. doi: 10.1016/j.visinf.2018.12.006 2

[50] V. Yoghourdjian, Y. Yang, T. Dwyer, L. Lawrence, M. Wybrow, and K. Marriott. Scalability of network visualisation from a cognitive load perspective. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1677–1687, Feb. 2021. doi: 10.1109/TVCG.2020.3030459 3