Graph Visualization Design Guidelines as Learnable Predicates

Sjoerd Vink[1,2,3][a], Brian Montambault[2][b], Mingwei Li[2][c], Remco Chang[2,3][d] and Michael Behrisch[1,3][e]

[1]*Information and computing sciences, Utrecht University, Utrecht, Netherlands*
[2]*Department of Computer Science, Tufts University, Medford, USA*
[3]*GraphPolaris, Utrecht, Netherlands*
{*s.a.vink, m.behrisch*}*@uu.nl*

Keywords: Graph visualization, predicate learning, visualization recommendation

Abstract: Graphs are widely used to represent complex, interconnected data across domains, yet choosing an effective visualization remains difficult because existing design knowledge is fragmented and inconsistent. This lack of a unified foundation prevents researchers from integrating findings into a cumulative body of knowledge, leaving valuable results isolated. It also hinders designers and practitioners, who cannot readily translate such findings into actionable strategies for their own goals and contexts. We propose a predicate-based representation that formalizes visualization guidelines as bounded conditions over descriptive graph statistics. Predicates directly mirror the qualitative structure of design guidelines. For example, a rule might specify that if graph density is low, a node-link diagram is appropriate, whereas if density is high, an adjacency matrix should be used. Unlike static handcrafted rules, they can also be learned, optimized, and adapted as new findings or usage contexts emerge. As a result, fragmented knowledge is consolidated into a formal and extensible foundation for graph visualization design and recommendation. We evaluate this approach by testing its ability to (i) recover expert rules and (ii) adapt to user-specific preferences while generalizing to unseen graphs. The results show that the learned predicates closely reproduce expert-derived guidelines, accommodate diverse preference patterns, and achieve strong performance on held-out data, demonstrating a promising path toward more systematic and cumulative graph visualization design knowledge.

1 INTRODUCTION

Graphs are a powerful way to represent complex, interconnected data across domains ranging from biology (Baitaluk et al., 2006) and social science (Zinovyev, 2010) to finance (Didimo et al., 2014) and cybersecurity (Noel et al., 2016). Yet, while graphs themselves are flexible and expressive, effectively visualizing them remains a persistent challenge (Li et al., 2023). Practitioners often rely on node-link diagrams rather than visualizations that may be more effective for a given task (Keller et al., 2006). In the remainder of this paper, we use "graph" to mean multivariate graphs unless otherwise specified.

To move beyond default choices like node-link diagrams, researchers have explored scenarios in which alternative visualization techniques are more effective. Ghoniem et al. (Ghoniem et al., 2004) found that node-link diagrams are more effective for smaller, sparser graphs, while adjacency matrices are more effective for larger, denser graphs. Similarly, Wang et al. (Wang et al., 2024) found that chord diagrams are well-suited for graphs with a medium number of nodes (albeit loosely defined) and moderate edge densities, providing a good balance between overview and readability. Nobre et al. (Nobre et al., 2019) took a step toward consolidating these findings and graph visualization guidelines into recommendations for practical use. However, these recommendations are presented in a static table that is not easily extensible, and do not cover all visualization designs (e.g., chord diagrams).

The limitations of current approaches based on fragmented design guidelines pose challenges for both researchers and practitioners. For researchers, the lack of a shared representation means that empirical findings are rarely incorporated into a common

[a] https://orcid.org/0009-0006-1094-3725
[b] https://orcid.org/0009-0000-1988-406X
[c] https://orcid.org/0000-0002-0457-8091
[d] https://orcid.org/0000-0002-6484-6430
[e] https://orcid.org/0000-0002-1102-103X

corpus, leaving valuable results isolated rather than building toward cumulative knowledge. For designers, the lack of a common foundation makes graph visualization design difficult. Current guidelines do not capture factors such as domain standards, preferences, and graphs with varying complexity and statistical properties. Moreover, it is often unclear whether a guideline derived for purely structural graphs generalizes to richer settings, such as multivariate graphs, weighted graphs, or multi-graphs. Without a formal foundation, existing design knowledge remains difficult to apply consistently and even harder to extend, leaving research and practice disjointed.

We make two contributions to bridge this gap: (i) a predicate-based representation of graph visualization design guidelines, and (ii) an algorithm for learning these predicates from data. Predicates mirror the structure of existing empirical rules and guidelines (see section 3 and section 5 for a more detailed explanation). To derive such rules, we propose a predicate induction algorithm that learns from a labeled dataset, where each graph is represented as a vector of graph statistics and the label indicates the appropriate visualization type, based either on established guidelines or user preferences. This algorithm not only learns but can also optimize and adapt predicates as new data or user feedback becomes available. This predicate-based representation captures empirical findings as explicit, human-readable rules that can be inspected, compared, and accumulated across studies. It also supports systematic refinement, since predicates can be evaluated, optimized, and updated as new evidence or feedback becomes available. Finally, the approach remains extensible, allowing new rules, graph features, and visualization types to be incorporated as the field evolves.

We evaluate our predicate-based representation and induction algorithm through two experiments, each corresponding to the usage scenarios introduced in section 6. First, we test the system's ability to recover known rules by examining whether predicates learned from a labeled dataset reproduce expert-defined guidelines from the literature. Following Nobre et al.'s reference table (Nobre et al., 2019), we label synthetically generated graphs according to their most appropriate visualization type and evaluate how closely the recovered predicates match these expert rules. Second, we evaluate personalization and recommendation by simulating different user preference profiles and testing how well the induced predicates generalize to held-out graphs. We generate labels that vary in their adherence to literature-derived guidelines and assess whether the learned predicates both adapt to these preferences and correctly recommend visualizations. Across both experiments, our approach successfully reconstructs expert rules, adapts to diverse usage patterns, and generalizes well on held-out data, demonstrating the potential of predicates as a foundation for accumulating and operationalizing graph visualization design knowledge.

2 RELATED WORK

Prior work can be grouped into two main areas: studies evaluating the effectiveness of graph visualizations, and systems that map data properties to visual encodings using either rules or machine learning.

2.1 Visualizing Graphs

Recent work on graphs in practice highlights that effective visualization is often indispensable for real-world analysis, where users rely on visual encodings to navigate complexity and extract insights (Li et al., 2023). Nobre et al. (Nobre et al., 2019) categorize graph visualizations into two high-level groups: explicit and implicit approaches. Schulz et al. (Schulz, 2011), focusing on tree visualizations, propose a taxonomy that instead distinguishes among explicit, implicit, and hybrid designs. Explicit approaches include node-link layouts (Battista et al., 1998), (Herman et al., 2000), (Schulz et al., 2013), (Nobre et al., 2019) and tabular representations (Becker et al., 1995), (Godsil and Royle, 2001), (Buono et al., 2021), (Bezerianos et al., 2010), (Longabaugh, 2012), while implicit approaches are common for trees and hierarchies (Johnson and Shneiderman, 1998), (Andrews and Heidegger, 1998), (Nobre et al., 2019). There are also hybrid alternatives available such as NodeTrix (Henry et al., 2007).

Researchers have devoted substantial effort to empirically evaluating when and why particular techniques are most effective, with the goal of distilling guidelines that can inform practice. For example, Ghoniem et al. (Ghoniem et al., 2004) and more recently Wang et al. (Wang et al., 2024) conducted controlled experiments demonstrating how the suitability of visualization methods depends on graph characteristics such as size and density. Purchase et al. (Purchase et al., 2002) examined the role of layout aesthetics, such as minimizing edge crossings, maintaining uniform edge lengths, and promoting symmetry, in improving graph readability. Similarly, Keller et al. (Keller et al., 2006) compared the effectiveness of matrices versus node-link diagrams in engineering design contexts, revealing domain-specific trade-offs. Beyond these focused experiments, Vehlow et al. (Vehlow et al., 2017) provided an empirical syn-

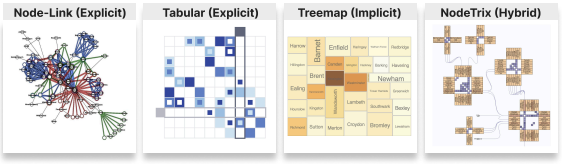

Figure 1: Examples of graph visualization techniques corresponding to the categories in subsection 2.1: node-link (Lambert et al., 2012) and adjacency matrix (Alper et al., 2013), Treemap (Slingsby et al., 2009), and NodeTrix (Henry et al., 2007).

thesis of techniques for visualizing communities and clustered graphs, consolidating findings across multiple studies.

Building on this line of work, Nobre et al. (Nobre et al., 2019) developed a reference table scoring visualization types against selected graph characteristics, such as number of nodes/edges, graph type, and attribute richness. Their study provided the main inspiration for ours, motivating us to formalize these rules in an extensible and learnable representation.

2.2 Systems for Mapping Data to Visualizations

Systems for mapping data to visualizations aim to operationalize design knowledge by linking data characteristics with appropriate visual encodings. We categorize prior work into two main paradigms (Hu et al., 2019): rule-based and machine-learning approaches.

Rule-Based Approaches: Rule-based systems specify mappings through handcrafted guidelines, often grounded in perceptual principles. For tabular data, a range of systems have been proposed (Mackinlay, 1986), (Casner, 1991), (Roth et al., 1994), (Bertin, 1983), (Cleveland and McGill, 1984), (Wongsuphasawat et al., 2015), (Wongsuphasawat et al., 2017), (Mackinlay et al., 2007), (Moritz et al., 2018) that rank or recommend visualizations based on perceptual and structural principles. For graph visualization specifically, only specialized rule-based tools exist. For example, TreePlus (Lee et al., 2006) and NetLens (Kang et al., 2007) focus on narrow topological cases such as tree-like structures or actor–group networks. The work of Nobre et al. (Nobre et al., 2019) introducing the scoring table also falls in this category. While valuable, this framework does not consider nuances such as directedness, weightedness, temporal or spatial dimensions, self-loops, or parallel edges, limiting its applicability across the full diversity of graph data. Also, fixed rules struggle to capture the complexity of graph data. Handcrafted mappings often rely on simple statistics such as node count or density, but they do not scale to multivariate graphs with attributes, weighted edges, or domain-specific structures (e.g., biological networks vs. social networks). Finally, as graphs vary along many interacting statistics, the combinatorial growth of rules becomes unmanageable. Capturing all relevant combinations by hand is infeasible, and conflicting prescriptions are difficult to reconcile without a learnable approach.

Machine-Learning Approaches: Machine-learning approaches address some limitations of handcrafted rules by learning mappings directly from data. For tabular visualization, a range of machine-learning and query-driven systems have explored automated recommendation (Hu et al., 2019), (Luo et al., 2018), (Li et al., 2021), (Wongsuphasawat et al., 2016), (Dibia, 2023), leveraging corpora, intent modeling, declarative queries, and large language models. To the best of our knowledge, no machine learning–based systems currently exist that map graphs to visualizations, and no corpus of graph–visualization mappings is available to train such models, which makes data-driven approaches particularly challenging.

In summary, prior work has established a broad repertoire of graph visualization techniques, empirical guidelines linking them to graph statistics, and systems that attempt to operationalize these mappings through rules or learning. Yet, these contributions remain fragmented. This gap motivates our contribution, namely a predicate-based representation that formalizes guidelines as explicit conditions over graph statistics, and an induction algorithm that learns these predicates from data.

3 DESIGN GUIDELINES AS PREDICATES

Rule-based and machine-learning approaches have both advanced automated visualization design, but each is limited for graph data: rule-based systems are interpretable yet brittle and hard to extend, while machine-learning systems adapt flexibly but often act as opaque black boxes. We address this gap by expressing graph visualization guidelines as bounded, interpretable predicate conditions over graph statistics. This section focuses on the representational role of predicates and how they consolidate design knowledge. section 4 then describes how they can be learned from labeled data.

3.1 Background: Predicates

A predicate is a logical condition that evaluates to `true` or `false` for a given object. Such conditions are typically expressed as functions that take an in-

put and return a Boolean value; i.e., for a number x, IsEven(x) returns `true` if x is even and `false` otherwise.

In our setting, predicates are built from clauses over graph statistics, where each statistic can be *continuous*, *binary*, or *categorical*. A clause constrains a given graph statistic x according to its type:

- **Continuous**: an interval constraint that evaluates to `true` for a graph G when $a < x < b$, and to `false` otherwise, where $a \in \mathbb{R}$ and $b \in \mathbb{R}$ are the parameters of the predicate.
- **Binary**: an equality constraint that evaluates to `true` when $x = a$, and `false` otherwise, where $a \in \{0,1\}$ is the parameter of the predicate.
- **Categorical**: a membership constraint that evaluates to `true` if $x \in a$ where a is a set of values and the parameter of the predicate.

For example, a continuous clause might specify that the graph's density lies within a given interval (*density* $\in [0,0.1]$). A binary clause could state that the graph is directed (*is-directed* $= 1$). A categorical clause might require that the graph type belongs to a subset of categories (*graph-type=tree*). A predicate is then the conjunction of one or more such clauses, and it evaluates to true if and only if all of its clauses evaluate to true. For instance, *(density* $\in [0,0.1] \wedge$ *is-directed* $= 1 \wedge$ *graph-type=tree)*, which evaluates to true only for graphs that are simultaneously sparse, directed, and a tree, in other words, a hierarchy.

This abstraction provides a natural way to capture design guidelines. Guidelines can be understood as statements about which kinds of graphs call for which design choices, and predicates supply the formal structure to express such statements in a clear and extensible manner. By encoding guidelines as explicit conditions over graph statistics, predicates can be evaluated to determine which guideline best fits a given graph. At the same time, the representation is modular, since new guidelines or visualization techniques can be incorporated by adding or adjusting predicate clauses rather than reworking the entire rule set. Finally, predicates preserve transparency, since each outcome can be traced back to explicit conditions, allowing researchers and practitioners to inspect and refine the reasoning behind design choices.

3.2 Graphs and Predicates in Feature Space

Each graph G can be described by a vector of statistics $s = (s_1, s_2, \ldots, s_M)$, where each s_j represents a graph property such as its density, number of nodes, or clustering coefficient. This vector embeds G as a point in

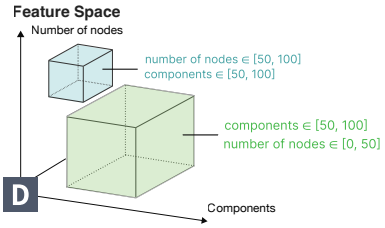

Figure 2: From predicates to feature space: each box corresponds to a bounded region of graph statistics that encodes a visualization guideline.

$\mathbb{R}^M$, which we call the *feature space* of graphs. The feature space provides a common coordinate system in which graphs can be compared, forming the basis for expressing design guidelines as predicates and for learning and refining them from data.

In this space, a predicate clause restricts one coordinate of the feature vector to an allowed set of values, while a conjunction of predicate clauses defines an axis-aligned region of the space. Visualization guidelines can then be expressed as such regions, since they specify conditions on which kinds of graphs are suited to which design choices. For example, when considering modularity and clustering coefficient, each graph is represented as a point in the two-dimensional plane. The guideline "*use NodeTrix for highly clustered and modular graphs*" then corresponds to a rectangle bounded by (modularity $\in [0.6,1.0] \wedge$ clustering coefficient $\in [0.5,1.0]$).

More elaborate guidelines can be expressed as unions or intersections of such regions, capturing conditions that overlap or exclude each other. Representing predicates in feature space makes the structure of design knowledge explicit. Guidelines that overlap correspond to alternative but compatible design choices, while disjoint regions signal potential conflicts. This geometric view also supports consolidation, because rules derived from different studies can be compared by examining how their regions align, and newly proposed rules can be integrated by adding regions to the space.

At the same time, the rigidity of hard cut-offs becomes apparent in this representation. Two graphs with nearly identical properties may fall on opposite sides of a boundary. For example, a hierarchical graph with 10 layers may satisfy a sunburst guideline, while one with 11 layers does not, even though they are nearly indistinguishable in practice. Such discontinuities underscore the limitations of hard, static boundaries and motivate a probabilistic extension. In section 4, we build on the feature-space representation to relax these hard cut-offs, allowing predicate boundaries to be learned and adapted from data.

4 PREDICATE INDUCTION ALGORITHM

While the predicate-based representation from the previous section makes design guidelines interpretable, manually curating predicates as design rules is insufficient. The landscape of graph visualization evolves with new findings, diverse datasets, and shifting user preferences. To remain useful, predicates must therefore be *induced* from data rather than only prescribed in advance. Predicate induction addresses this by making the axis-aligned regions learnable.

Learning Predicate Regions: Instead of fixing regions a priori, we optimize them to align with labeled training examples. Building on the approach of Montambault et al. (Montambault et al., 2024), we formalize predicate induction through two components: a probability function, which evaluates how well a graph satisfies a candidate predicate, and a loss function, which guides the adjustment of predicate intervals. Together, these elements define an optimization procedure that refines predicates into accurate and generalizable design rules.

The Probability Function: For each visualization design guideline v, we define a candidate predicate Φ_v as a conjunction of bounded conditions over graph statistics. Formally, let $\Phi_v = (\phi_1, \phi_2, \ldots, \phi_M)$ be the parameters of learnable predicate clauses for M graph statistics. Each clause $\phi_j := (\mu_j, r_j)$ specifies a learnable region for the j-th statistic. We write $s_i = (s_{i1}, s_{i2}, \ldots, s_{iM})$ for the statistics vector of a given graph i, with s_{ij} denoting its value on the j-th statistic. The induction algorithm then evaluates how well s_i satisfies Φ_v through a probability function:

$$\Pr(\Phi_v, s_i) := \frac{1}{1 + \sum_{j=1}^{M} \mathscr{L}(\phi_j, s_{ij})} \tag{1}$$

Loss Functions by Data Type: The loss function $\mathscr{L}(\phi_j, s_{ij})$ specifies the mismatch between the value s_{ij} and the clause ϕ_j. We define a loss function for each data type represented in the graph statistics: continuous, binary, and categorical. For continuous statistics, e.g., *number of nodes or edges*, we define a differentiable bump function:

$$\mathscr{L}_{continuous}(\phi_j, s_{ij}) = \left|\frac{1}{r_j} \cdot (s_{ij} - \mu_j)\right|^b$$

where b controls the steepness and roundness of the bump (empirically, we set $b = 3$). This function gives low loss (high probability) to values near the midpoint and penalizes those farther away.

For binary statistics, e.g., graph statistics such as *is-directed* or *is-multi-graph*, we denote the learnable parameter, sensitivity for the binary variable, as $\phi_j := (w_j)$. The loss is defined as:

$$\mathscr{L}_{binary}(\phi_j, s_{ij}) = \exp(-s_{ij} \cdot w_j)$$

where $s_{ij} \in \{-1, 1\}$ is the binary value, and w_j is a learned weight. A positive w_j favors $s_{ij} = 1$ in that it reduces the loss, while a negative w_j favors $s_{ij} = -1$.

For categorical statistics, e.g., *graph-type* $\in$ [Tree, Sparse], the learnable parameters for the clause is list of $\phi_j := (w_{j,c_1}, w_{j,c_2}, \ldots)$. We define loss as the average binary loss over all categorical values:

$$\mathscr{L}_{categorical}(\phi_j, s_{ij}) = \exp(-\sum_{c \in C_j} s_{ij,c} \cdot w_j)$$

where C_j is the set of all possible values of the j-th statistic, and $|C_j|$ its cardinality. $w_{j,c}$ is a learned weight for a given value c in C_j. $s_{ij,c}$ is used to denote a binary value, where $s_{ij,c} = 1$ if $s_{ij} = c$ and $s_{ij,c} = -1$ otherwise. In this way, categorical variables are treated similarly to numbers, preserving consistency in our probabilistic framework.

Optimization Objective: Let S denote the set of graph statistics vectors over a set of graph instances. Y_v is a vector of length $|S|$ representing the labels of a visualization design v (e.g., v = node-link) such that $y_{v,i}$ is a binary value denoting if graph s_i is labeled as v. Given the graphs S and the labels Y_v, we compute an optimal predicate for the visualization v by minimizing the binary cross entropy:

$$\begin{aligned}\mathscr{L}(\Phi_v \mid S, Y_v) = \frac{1}{N} \sum_{i=1}^{N} & y_{v,i} \log(\Pr(\Phi_v, s_{ij})) \\ & + (1 - y_{v,i}) \log\left(1 - \Pr(\Phi_v, s_{ij})\right)\end{aligned} \tag{2}$$

This optimization adjusts the centers and ranges of each predicate's intervals in continuous graph statistics, as well as their sensitivity in binary and categorical cases, to ensure that the probability scores align with the labeled examples. Each optimized region in feature space then corresponds to a learned visualization guideline. For instance, if the training data consistently labels sparse graphs with fewer than 200 nodes as "node-link," the induction algorithm will converge on a predicate approximating (*density* $\in [0, 0.1] \wedge$ *node count* $\in [0, 200]$).

By learning predicates from labeled examples, our approach moves beyond fixed, hand-crafted rules. The induction algorithm produces guidelines that remain interpretable as ranges on graph statistics, but that also adapt to new evidence, user preferences, or domain-specific contexts. This flexibility makes predicates both a stable representation of guidelines and a mechanism for continuous refinement.

5 PREDICATES APPLIED TO GRAPH VISUALIZATION

We now apply predicates to graph visualization by defining the graph statistics that form the input space and examining how predicate-based guidelines behave on graphs. This illustrates how the earlier abstract representation becomes concrete in practice.

5.1 Input Space

To determine graph statistics that comprise the input space, we began with the 168 references compiled by Nobre et al. (Nobre et al., 2019) and extended this corpus with papers published in the past five years in TVCG and CHI. Our initial search yielded 1,680 papers in TVCG and 2,200 in CHI containing the keyword "graph." From this set, we retained only those that explicitly articulated design guidelines or employed a graph statistic as a central element in the design or evaluation of a visualization technique. After applying these criteria, the corpus comprised 193 papers. From this refined set, we systematically extracted the graph statistics identified by authors as influential for visualization design and effectiveness. We also incorporated a small set of additional graph metrics based on the authors' domain knowledge, ensuring coverage of statistics commonly used in practice but underrepresented in the surveyed literature.

The resulting input space (see Table 1) reflects the statistics most often used to inform visualization choices in the literature and in practice. For clarity, we organize these statistics into four broad categories that frequently recur in the literature: general properties, connectivity measures, cohesion measures, and elements. These statistics serve as the features over which our induction algorithm learns predicate regions, enabling guidelines to be expressed, refined, and compared systematically.

Category	Measure	Data Type
General	Graph type	`[Tree, Cycle, ...]`
	Directed	`Boolean`
	Spatial	`Boolean`
	Planar	`Boolean`
	Hypergraph	`Boolean`
	Layer count	`Numeric (integer)`
Connectivity	Self-loops	`Numeric (integer)`
	Parallel edges	`Numeric (integer)`
	Size (nodes and edges)	`Numeric (integer)`
	Components	`Numeric (integer)`
	Connectivity / paths	`Numeric (float)`
	Diameter	`Numeric (integer)`
	Average path length	`Numeric (float)`
Cohesion	Cut ratio	`Numeric (0, 1)`
	Separability	`Numeric (0, 1)`
	Density	`Numeric (0, 1)`
	Cluster density	`Numeric (0, 1)`
	Clustering coefficient	`Numeric (0, 1)`
	Triangle count	`Numeric (float)`
	Modularity	`Numeric (0, 1)`
	Average degree	`Numeric (float)`
	Centrality measures	`Numeric (float)`
	Communities	`Numeric (integer)`
	Clusters	`Numeric (integer)`
Elements	Node/Edge types	`Numeric (integer)`
	Node/Edge attributes	`Numeric (integer)`

Table 1: Graph statistics that define the input feature space.

5.2 Uncertainty and Trade-offs in Guidelines

Visualization design guidelines are rarely absolute. Graphs often fall near the boundaries of statistical ranges, and different guidelines may overlap or even conflict. Moreover, empirical studies frequently disagree on the exact thresholds, underscoring the need for a representation that tolerates uncertainty.

For example, consider three graphs with density values of 0.05, 0.1, and 0.15. A deterministic predicate such as $\textit{density} \in [0, 0.1]$ applied to these three groups would evaluate to true, true, and false, respectively. This draws a hard boundary between those inside and outside the range, but misses important nuance. The first graph lies in the center of the interval and strongly satisfies the guideline, the second sits right on the boundary, and the third is just outside the range and often nearly indistinguishable from the former in practice (note: the range of *density* is from 0 to 1). By interpreting predicate membership probabilistically (see Equation 1), these cases are differentiated: the first graph receives the highest probability, the second moderate, and the third minimal, but nonzero, probability, despite having a graph statistics that lie outside of the predicate's interval.

Applied to graph visualization, this treatment of uncertainty has several implications. First, it clarifies trade-offs in overlapping regions of the feature space where a single graph meets the predicates of more than one visualization guideline. That is, when its statistics satisfy multiple conjunctive conditions simultaneously, making it eligible for several visualization types at once. Second, it reduces the brittleness of fixed thresholds. For example, values in guidelines like "*greater than 200 nodes*" or "*density below 0.1*" become zones of gradual transition rather than

hard cut-offs. Third, it provides a basis for comparing guidelines across studies, as overlapping regions reveal consensus while divergences highlight conflicts or gaps. Finally, it creates opportunities for adaptive or personalized visualization. When multiple guidelines receive partial support, their relative probabilities can guide recommendations while keeping conditions interpretable. In this way, predicates encode graph visualization guidelines while also revealing their uncertainty, overlaps, and points of contention.

6 USAGE SCENARIOS

We now illustrate how predicates can be used in practice through two usage scenarios: recovering implicit rules from labeled data and adapting predicates to user preferences. section 7 evaluates these scenarios by testing rule recovery and assessing how well learned predicates adapt to user preferences and generalize to unseen graphs.

6.1 Rule Recovery

A major advantage of the predicate-based representation is its ability to recover design rules directly from labeled data rather than relying on manually encoded guidelines, as in systems like Draco (Moritz et al., 2018). Given a dataset of graphs labeled with their designated visualization type, each graph is mapped to a feature vector of statistics, and the learning process induces a predicate for each visualization. Each clause specifies a constraint on a statistic, and optimization adjusts these parameters so that $\Pr(\Phi_v, s_i)$ (Equation 1) aligns with the observed labels, converging on regions of the feature space that best separate visualization choices. For example, if graphs with high modularity and clustering coefficient are consistently visualized with NodeTrix diagrams, the learning process would recover a predicate such as *modularity* $\in$ *[0.6,1.0]* $\wedge$ *clustering coefficient* $\in$ *[0.5,1.0]*, capturing conditions under which hybrid visualizations are preferred.

6.2 Personalization and Recommendation

Predicates not only recover rules from data but also support personalization and recommendation. Because each guideline is expressed as a conjunction of clauses $\Phi_v = (\phi_1, \ldots, \phi_M)$, user or domain preferences can be incorporated by adjusting clause intervals: positive feedback expands relevant ranges, while negative feedback contracts them. This yields interpretable refinements, in contrast to prior personalized systems (Gotz and Zhou, 2009), (Mutlu et al., 2016), (Ottley et al., 2015), (Qian et al., 2022), and allows separate predicate sets $\{\Phi_v\}$ to reflect different user groups or contexts. The same structure enables visualization recommendation for new graphs. Mapping a graph G to a feature vector s and evaluating $\Pr(\Phi_v, s)$ produces a probability distribution over visualization types that reflects how well the graph satisfies each guideline. Graphs deep inside a predicate's region receive strong support, while those near boundaries may partially satisfy several guidelines, making trade-offs explicit. Because recommendations derive directly from clause contributions (e.g., size, density, clustering), they remain fully interpretable. Treating personalization and recommendation as two uses of the same probabilistic evaluation unifies user adaptation and design guidance in a transparent, extensible model.

7 EVALUATION

We evaluate our predicate representation and induction algorithm through two structured experiments:

1. **Rule recovery** – testing whether the induction algorithm can reconstruct known guidelines when only labeled data is provided.
2. **Personalization and recommendation** – evaluating how predicates adapt to user-specific preferences and how well these personalized predicates generalize to unseen graphs.

Evaluation Data We generated a dataset of 1200 sample graphs, consisting of 80% sparse and dense graphs, 10% cycles, and 10% trees. Graph sizes range from 2 to 200 nodes, and include both simple and multivariate graphs. Graph statistics were extracted for all graphs to produce feature vectors. We use 1000 graphs for training (rule recovery and personalization) and reserve 200 graphs for evaluating generalization.

7.1 Recovering Rules from Labeled Data

This experiment corresponds to the *rule recovery* scenario. Here, the induction algorithm is trained solely on labeled graph–visualization pairs, without being given any predefined predicates or expert rules. The aim is to test whether the algorithm can infer meaningful predicates directly from the data, mimicking a situation where only empirical evidence is available. If the learned predicates match the original ones used to generate the labels, we can conclude that the in-

duction algorithm has successfully recovered the underlying design rules.

Generating Ground-truth: To establish an impartial ground-truth dataset that maps each generated graph to a visualization recommendation, we refer to Nobre et al. (Nobre et al., 2019)'s reference table. In their scheme, a score of zero indicates no support and a score of three indicates strong support. To make these recommendations operational, we expressed each "full support" condition (score = 3) as a predicate clause over the corresponding statistic. When multiple statistics were mentioned together, we combined them conjunctively into a single predicate. Applying these predicates to our generated graphs allowed us to assign each graph the visualization type it best supports. This procedure produced the ground-truth labels for our feature vectors. Notably, their guidelines seldom, if ever, recommend the use of adjacency matrices, treemaps, and sunbursts. The predicates are shown in Table 2.

Visualization	Predicate
Node-Link	n_nodes $\in [0,100]$, graph_type $\in [sparse, k-partite, tree]$, node_types $\in [1,1]$, and edge_types $\in [1,1]$
Attribute-driven positioning	n_nodes $\in [0,100]$, graph_type $\in [sparse, k-partite]$, node_attributes $\in [0,5]$, and node_types $\in [1,5]$
Attribute-driven faceting	n_nodes $\in [0,100]$, graph_type $\in [sparse]$, node_attributes $\in [0,5]$, and node_types $\in [1,1]$
Adjacency Matrix	n_nodes $\in [0,100]$, graph_type $\in [dense]$, node_attributes $\in [5,10]$, node_types $\in [1,1]$, edge_attributes $\in [0,3]$, and edge_types $\in [1,1]$
Quilts	n_nodes $\in [0,100]$, graph_type $\in [sparse, k-partite, tree]$, node_attributes $\in [0,10]$, node_types $\in [1,5]$, edge_attributes $\in [0,10]$, and edge_types $\in [1,1]$
BioFabric	n_nodes $\in [0,100]$, graph_type $\in [sparse, dense]$, node_attributes $\in [0,10]$, node_types $\in [1,5]$, edge_attributes $\in [0,10]$, and edge_types $\in [1,5]$
Treemap	graph_type $\in [sparse, tree]$, node_attributes $\in [0,5]$, and node_types $\in [1,1]$
Sunburst	graph_type $\in [sparse, tree]$, node_attributes $\in [0,5]$, and node_types $\in [1,1]$

Table 2: Nobre et al.'s multivariate graph visualization design guidelines (Nobre et al., 2019) converted to predicates.

Reconstructed Predicates To assess the ability to recreate the guidelines as predicates, we compare the learned predicates with the Nobre et al. (Nobre et al., 2019) guidelines by evaluating the degree of overlap for each predicate clause, measured using their intersection over union (IOU). The values of IOU range from 0 to 1, where 0 represents low accuracy where there is no overlap between the ground-truth and the learned predicates, and 1 means high accuracy with complete overlap between the two. Average IOU for each visualization (Table 3) indicate that, in most cases, the predicates learned closely match the ground truth. The predicates defined for the adjacency matrix and sunburst visualization types could not be recovered, since Nobre et al.'s guidelines never recommended these techniques when only considering scores of three, and therefore they did not appear in the labeled dataset. Overall, the general match between the ground-truth and the learned predicates demonstrates that the induction algorithm can induce visualization guidelines from data.

Visualization	Average IOU (higher better)
Node-Link	0.83
Attribute-driven positioning	0.55
Attribute-driven faceting	0.79
Quilts	0.65
BioFabric	0.31
TreeMap	0.66

Table 3: Average IOU for each visualization defined by Nobre et al. (Nobre et al., 2019).

7.2 Personalization and Recommendation

This experiment corresponds to the *personalization and recommendation* scenario. Here, we evaluate whether the system can adapt to user-specific preferences and generalize the learned predicate boundaries to unseen graphs. We test whether the algorithm can flexibly adapt when users deviate from given rules. By simulating different adherence levels, we assess the robustness of the induction to noisy or inconsistent feedback.

We base these labels on a set of predicates derived from prior literature (see section 9), which captures common guidelines. While not exhaustive, this collection spans the major visualization types studied in the literature and provides a representative baseline of design rules for our evaluation. To simulate different behaviors, we generate three user profiles: an *informed user* who follows expert predicates consistently, a *semi-informed user* who deviates 25% of the time, and an *uninformed user* who deviates 50% of the time. These profiles produce three versions of training labels. Running predicate induction on

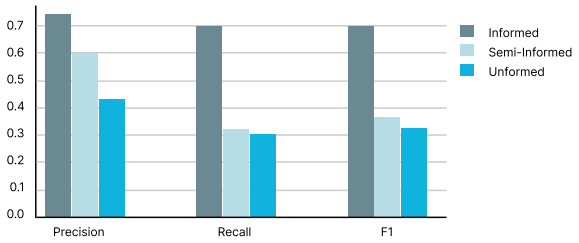

Figure 3: Generalization performance across user types.

each labeled dataset produces one learned predicate for every visualization type. For a given graph, the algorithm assigns a score to each visualization type based on how well the graph satisfies its corresponding predicate. The visualization type with the highest score is recommended as the label. If two types receive the same score, we break the tie by choosing the type that appeared more frequently in the training labels. After inducing predicates from each user's labeled data, we evaluate recommendation performance on the 200 held-out graphs. A prediction is considered correct when the visualization with the highest probability matches the simulated user's label.

As shown in Figure 3, predicates learned from consistent user behavior generalize well to unseen data, achieving an F1-score of 0.70. Performance declines as user behavior becomes noisier, but the system still captures broad patterns in preference and applies them to new graphs. These results highlight that personalization and recommendation emerge naturally from the same predicate-based mechanism.

8 DISCUSSION

While the predicate-based representation and the mechanism for learning predicates provide a foundation for consolidating graph visualization design knowledge, several limitations highlight opportunities for future work. At the same time, these contributions open pathways toward more systematic, interpretable, and cumulative progress in graph visualization research and practice.

Expressiveness of the Predicate Representation: In the present formulation, each predicate is treated as an independent, conjunctive rule defining a single bounded region of the feature space. This structure improves interpretability but limits expressiveness. Dependencies between conditions, such as density thresholds that differ for directed versus undirected graphs, cannot be modeled. Likewise, many visualization types may be appropriate under multiple disjoint conditions, which a single predicate region cannot capture. Extending the representation to support conditional or disjunctive rules would make the model more expressive. Additionally, our approach maps graphs to visualization types, maintaining tractability but omitting encoding-level considerations such as layout aesthetics, perceptual constraints, and interaction techniques. Extending predicates to capture these lower-level guidelines would expand their applicability and better support design decisions. Applying this approach to tabular data, is more difficult, since tabular visualizations often rely on transformations like grouping or binning, which cannot be reduced to simple bounded conditions.

Integrating Expert Knowledge with Learned Rules: The current system focuses on learning predicates from labeled data. Incorporating expert-derived guidelines as priors, then refining them through feedback, would combine the stability of established knowledge with the adaptability of data-driven refinement. Such integration would support cumulative knowledge building rather than treating expert rules and learned rules as separate resources.

Balancing Multiple Valid Explanations: For many visualization decisions, more than one predicate may offer a valid explanation. The present approach converges on a single predicate per visualization type, simplifying outcomes but not fully reflecting the diversity of possible design rationales. Future work could incorporate weighting or complexity penalties to balance multiple competing explanations, favor simpler predicates, or highlight alternative interpretations when several explanations hold.

Toward Systematic and Cumulative Design Knowledge: A key challenge for the field is the absence of systematic methods to compare graph-visualization techniques, assess their effectiveness across graph types and tasks, or determine when new designs are warranted. A shared repository of graph–visualization mappings would support synthesis, reproducibility, and large-scale learning, while clarifying the relative importance of the graph statistics used in practice. Identifying which features best predict visualization effectiveness would advance evidence-based, generalizable guidelines. By expressing such guidelines as predicates and refining them through learned adjustments, this work provides a structured framework that unifies interpretability, adaptability, and transparency, which helps consolidate dispersed design knowledge into a cumulative, coherent foundation.

9 CONCLUSION

This paper offers two contributions: a predicate-based representation that formalizes visualization guidelines as bounded, interpretable conditions over graph statistics, and a learning mechanism that refines these

predicates from labeled data or user feedback. Together, these components enable systematic comparison, extension, and consolidation of design knowledge. Looking forward, we will expand the predicate space to include additional statistics, visualization types, and conditional logic. With such advances, predicates could evolve into a unifying resource that supports both visualization research and evidence-based practice.

REFERENCES

Alper, B., Bach, B., Henry Riche, N., Isenberg, T., and Fekete, J.-D. (2013). Weighted graph comparison techniques for brain connectivity analysis. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 483–492.

Andrews, K. and Heidegger, H. (1998). Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs. In *IEEE Information Visualization Symposium*, pages 9–12.

Baitaluk, M., Sedova, M., Ray, A., and Gupta, A. (2006). Biologicalnetworks: visualization and analysis tool for systems biology. *Nucleic acids research*, 34(suppl_2):W466–W471.

Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. G. (1998). *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall PTR.

Becker, R. A., Eick, S. G., and Wilks, A. R. (1995). Visualizing network data. *IEEE Transactions on visualization and computer graphics*, 1(1):16–28.

Behrisch, M., Bach, B., Henry Riche, N., Schreck, T., and Fekete, J.-D. (2016). Matrix reordering methods for table and network visualization. In *Computer Graphics Forum*, volume 35, pages 693–716. Wiley Online Library.

Bertin, J. (1983). *Semiology of graphics*. University of Wisconsin press.

Bezerianos, A., Dragicevic, P., Fekete, J.-D., Bae, J., and Watson, B. (2010). Geneaquilts: A system for exploring large genealogies. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1073–1081.

Buono, P., Ceriani, M., and Costabile, M. F. (2021). Hypergraph data analysis with paohvis. In *SEBD*, pages 160–167.

Casner, S. M. (1991). Task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics (ToG)*, 10(2):111–151.

Cleveland, W. S. and McGill, R. (1984). Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554.

Dibia, V. (2023). Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. *arXiv preprint arXiv:2303.02927*.

Didimo, W., Liotta, G., and Montecchiani, F. (2014). Network visualization for financial crime detection. *Journal of Visual Languages & Computing*, 25(4):433–451.

Ghoniem, M., Fekete, J.-D., and Castagliola, P. (2004). A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE symposium on information visualization*, pages 17–24. Ieee.

Ghoniem, M., Fekete, J.-D., and Castagliola, P. (2005). On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information visualization*, 4(2):114–135.

Godsil, C. and Royle, G. F. (2001). *Algebraic graph theory*, volume 207. Springer Science & Business Media.

Gotz, D. and Zhou, M. X. (2009). Characterizing users' visual analytic activity for insight provenance. *Information Visualization*, 8(1):42–55.

Henry, N., Fekete, J.-D., and McGuffin, M. J. (2007). Nodetrix: a hybrid visualization of social networks. *IEEE transactions on visualization and computer graphics*, 13(6):1302–1309.

Herman, I., Melançon, G., and Marshall, M. S. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Trans. on Visualization & computer graphics*, 6(1):24–43.

Hu, K., Bakker, M. A., Li, S., Kraska, T., and Hidalgo, C. (2019). Vizml: A machine learning approach to visualization recommendation. pages 1–12.

Johnson, B. and Shneiderman, B. (1998). Tree-maps: A space filling approach to the visualization of hierarchical information structures. Technical report, UM Computer Science Department; CS-TR-2657.

Kang, H., Plaisant, C., Lee, B., and Bederson, B. B. (2007). Netlens: iterative exploration of content-actor network data. *Information Visualization*, 6(1):18–31.

Keller, R., Eckert, C. M., and Clarkson, P. J. (2006). Matrices or node-link diagrams: which visual representation is better for visualising connectivity models? *Information Visualization*, 5(1):62–76.

Kerren, A., Purchase, H., and Ward, M. O. (2014). *Multivariate Network Visualization: Dagstuhl Seminar# 13201, Dagstuhl Castle, Germany, May 12-17, 2013, Revised Discussions*, volume 8380. Springer.

Lambert, A., Queyroi, F., and Bourqui, R. (2012). Visualizing patterns in node-link diagrams. In *2012 16th International Conference on Information Visualisation*, pages 48–53. IEEE.

Lee, B., Parr, C. S., Plaisant, C., Bederson, B. B., Veksler, V. D., Gray, W. D., and Kotfila, C. (2006). Treeplus: Interactive exploration of networks with enhanced tree layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426.

Li, H., Appleby, G., Brumar, C. D., Chang, R., and Suh, A. (2023). Knowledge graphs in practice: Characterizing their users, challenges, and visualization opportunities. *IEEE Transactions on Visualization and Computer Graphics*.

Li, H., Wang, Y., Zhang, S., Song, Y., and Qu, H. (2021). Kg4vis: A knowledge graph-based approach for visualization recommendation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):195–205.

Longabaugh, W. J. (2012). Combing the hairball with biofabric: a new approach for visualization of large networks. *BMC bioinformatics*, 13:1–16.

Luo, Y., Qin, X., Tang, N., and Li, G. (2018). Deepeye: Towards automatic data visualization. In *2018 IEEE 34th international conference on data engineering (ICDE)*, pages 101–112. IEEE.

Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *Acm Transactions On Graphics (Tog)*, 5(2):110–141.

Mackinlay, J., Hanrahan, P., and Stolte, C. (2007). Show me: Automatic presentation for visual analysis. *IEEE transactions on visualization and computer graphics*, 13(6):1137–1144.

Montambault, B., Appleby, G., Rogers, J., Brumar, C. D., Li, M., and Chang, R. (2024). Dimbridge: Interactive explanation of visual patterns in dimensionality reductions with predicate logic. *IEEE Transactions on Visualization and Computer Graphics*.

Moritz, D., Wang, C., Nelson, G. L., Lin, H., Smith, A. M., Howe, B., and Heer, J. (2018). Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE transactions on visualization and computer graphics*, 25(1):438–448.

Mutlu, B., Veas, E., and Trattner, C. (2016). Vizrec: Recommending personalized visualizations. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 6(4):1–39.

Nobre, C., Meyer, M., Streit, M., and Lex, A. (2019). The state of the art in visualizing multivariate networks. In *Computer Graphics Forum*, volume 38, pages 807–832. Wiley Online Lib.

Noel, S., Harley, E., Tam, K. H., Limiero, M., and Share, M. (2016). Cygraph: graph-based analytics and visualization for cybersecurity. In *Handbook of statistics*, volume 35, pages 117–167. Elsevier.

Ottley, A., Yang, H., and Chang, R. (2015). Personality as a predictor of user strategy: How locus of control affects search strategies on tree visualizations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3251–3254.

Purchase, H. C., Carrington, D., and Allder, J.-A. (2002). Empirical evaluation of aesthetics-based graph layout. *Empirical Software Engineering*, 7:233–255.

Qian, X., Rossi, R. A., Du, F., Kim, S., Koh, E., Malik, S., Lee, T. Y., and Ahmed, N. K. (2022). Personalized visualization recommendation. *ACM Transactions on the Web (TWEB)*, 16(3):1–47.

Roth, S. F., Kolojejchick, J., Mattis, J., and Goldstein, J. (1994). Interactive graphic design using automatic presentation knowledge. pages 112–117.

Scheibel, W., Trapp, M., Limberger, D., and Döllner, J. (2020). A taxonomy of treemap visualization techniques. In *VISIGRAPP (3: IVAPP)*, pages 273–280.

Schöttler, S., Yang, Y., Pfister, H., and Bach, B. (2021). Visualizing and interacting with geospatial networks: A survey and design space. In *Computer Graphics Forum*, volume 40, pages 5–33. Wiley Online Library.

Schulz, H.-J. (2011). Treevis. net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15.

Schulz, H.-J., Nocke, T., Heitzler, M., and Schumann, H. (2013). A design space of visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375.

Slingsby, A., Dykes, J., and Wood, J. (2009). Configuring hierarchical layouts to address research questions. *IEEE transactions on visualization and computer graphics*, 15(6):977–984.

Vehlow, C., Beck, F., and Weiskopf, D. (2017). Visualizing group structures in graphs: A survey. In *Computer Graphics Forum*, volume 36, pages 201–225. Wiley Online Library.

Wang, K., He, S., Wang, W., Yu, J., Liu, Y., and Yu, L. (2024). Chordination: Evaluating visual design choices in chord diagrams for network data. *arXiv preprint arXiv:2408.02268*.

Wongsuphasawat, K., Moritz, D., Anand, A., Mackinlay, J., Howe, B., and Heer, J. (2015). Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics*, 22(1):649–658.

Wongsuphasawat, K., Moritz, D., Anand, A., Mackinlay, J., Howe, B., and Heer, J. (2016). Towards a general-purpose query language for visualization recommendation. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pages 1–6.

Wongsuphasawat, K., Qu, Z., Moritz, D., Chang, R., Ouk, F., Anand, A., Mackinlay, J. D., Howe, B., and Heer, J. (2017). Voyager 2: Augmenting visual analysis with partial view specifications. In *CHI*, pages 2648–2659. ACM.

Zinovyev, A. (2010). Data visualization in political and social sciences. *arXiv preprint arXiv:1008.1188*.

APPENDIX

Literature Predicate Set

Table 4 provides a sample of graph visualization guidelines expressed in predicate form. This set, derived from prior studies, is not comprehensive but demonstrates the idea.

Visualization	Predicate(s)	Derived from	Explanation
Node-Link	density $\in [0, 0.1]$, is_directed $\in [0.5, 1]$, self_loops $\in [0, 50]$	(Ghoniem et al., 2005)	Suitable for low-density directed graphs with some self-loops.
	components $\in [1, 5]$, avg_degree $\in [1, 3]$, clustering_coefficient $\in [0.1, 0.4]$	(Ghoniem et al., 2005)	Effective for graphs with a few components and low connectivity.
	node_types $\in [1, 3]$, edge_types $\in [1, 2]$, eccentricity $\in [0, 5]$	(Ghoniem et al., 2005) (Kerren et al., 2014)	Good for simple node and edge types and limited eccentricity.
Matrix	density $\in [0.1, 1]$, avg_degree $\in [10, 50]$, modularity $\in [0.3, 0.7]$	(Ghoniem et al., 2004) (Behrisch et al., 2016)	Can handle higher densities and modular structures.
	betweenness_centrality $\in [0.2, 0.5]$, eigenvector_centrality $\in [0.2, 0.8]$	(Ghoniem et al., 2005)	Suitable for spotting highly connected nodes.
	node_attributes $\in [2, 10]$, edge_attributes $\in [1, 5]$	(Ghoniem et al., 2005)	Suitable for graphs with numerous attributes.
NodeTrix	communities $\in [4, 10]$, clustering_coefficient $\in [0.5, 1]$, density $\in [0.1, 0.5]$	(Henry et al., 2007)	Ideal for modular, highly clustered graphs.
	node_types $\in [2, 5]$, modularity $\in [0.3, 0.8]$, avg_degree $\in [5, 15]$	(Henry et al., 2007)	Useful for graphs with moderate modularity and average degree.
	node_attributes $\in [3, 10]$, edge_types $\in [1, 3]$	(Henry et al., 2007)	Works well with graphs featuring moderate attribute diversity.
Node-Link Map	is_spatial $\in [0.5, 1]$, is_directed $\in [0, 1]$, avg_degree $\in [1, 5]$	(Schöttler et al., 2021)	Excellent for spatially relevant graphs with limited density.
	components $\in [1, 5]$, degree_assortativity $\in [-0.5, 0.5]$	(Schöttler et al., 2021)	Useful for few-component graphs with low degree variance.
PaohVis	n_nodes $\in [50, 500]$, node_types $\in [3, 6]$, edge_types $\in [2, 5]$	(Buono et al., 2021)	Effective for medium-sized graphs with multiple node and edge types.
	density $\in [0.05, 0.2]$, avg_degree $\in [5, 10]$, transitivity $\in [0.2, 0.6]$	(Buono et al., 2021)	Suitable for medium-density graphs.
Chord Diagram	n_nodes $\in [0, 6]$, edge_types $\in [1, 3]$, clustering_coefficient $\in [0.3, 0.7]$	(Wang et al., 2024)	Most effective for small graphs with low clustering.
	components $\in [1, 2]$, avg_degree $\in [2, 4]$, parallel_edges $\in [0, 5]$	(Wang et al., 2024)	Works best with a few components and minimal parallel edges.
Treemap	graph_type $\in [0.5, 1.5]$, modularity $\in [0.5, 1]$, n_nodes $\in [50, 200]$	(Scheibel et al., 2020)	Ideal for hierarchical or tree structures with medium node count.
	node_attributes $\in [5, 20]$, edge_attributes $\in [0, 2]$	(Scheibel et al., 2020)	Useful for visualizing attribute-rich nodes with minimal edge attributes.
	components $\in [1, 1]$, is_spatial $\in [0, 1]$	(Scheibel et al., 2020)	Effective for single-component, spatially relevant graphs.

Table 4: Our predicate mapping of existing graph visualizations to design guidelines.