NOVA: Knowledge-Augmented Visual Analysis for Automated Graph Layout Optimization

Nuo Xu* Tong Li† Siyu Mao Chenze Li Qi Jiang Xueqian Zheng Yunchao Wang
Guodao Sun Ronghua Liang

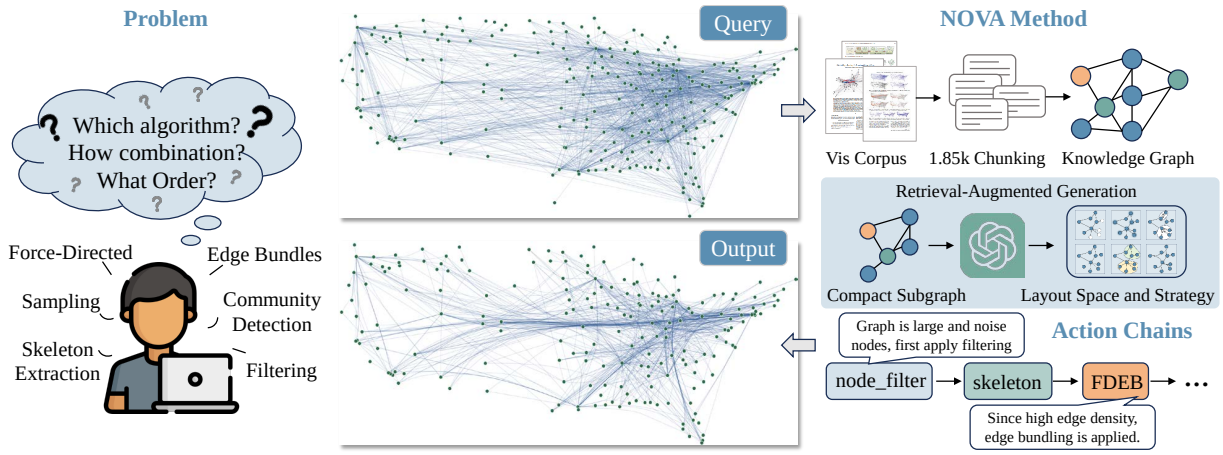

Figure 1: NOVA Framework. Intelligent large-scale graph layout optimization via visualization knowledge and structured RAG.

Abstract

Traditional graph layout optimization relies heavily on manual experience and lacks knowledge-driven mechanisms for perception, decision-making, and feedback when selecting and ordering layout operations for different graph characteristics and visual objectives, making it difficult to meet the growing demand for intelligent visual analysis. We propose NOVA, an intelligent graph layout optimization framework. NOVA leverages a Knowledge Graph-based RAG method to achieve multi-objective layout optimization tailored to large-scale graph characteristics. Built upon this framework, we also develop an interactive visual analysis system. Comparative experiments and user studies demonstrate that our method improves visual layout quality by an average of 16.95% relative to the initial layout, while enhancing the user experience compared to traditional layout tools.

Index Terms: Visualization, Graph Layout, RAG.

1 Introduction

Graph data are ubiquitous in real-world domains such as social networks, transportation systems, and bioinformatics [33]. However, real-world graphs vary greatly in scale and topological structure, ranging from sparse flow graphs with dozens of nodes to dense social networks with hundreds of thousands of nodes, which poses substantial challenges for graph layout visualization. To clearly depict graph structures within limited two-dimensional space, researchers have developed various layout optimization methods. Force-directed layouts [14, 17] model graphs as physical systems and iteratively converge to stable layouts via energy minimization; edge bundling techniques [22, 18, 23] aggregate spatially proximate edges to reduce visual clutter; graph sampling and coarsening methods [28, 46] compress large-scale graphs to accelerate layout computation and improve readability.

Graph layout optimization is inherently knowledge-intensive. For graphs with different topological properties, researchers must jointly consider graph size, edge density, connectivity, and other structural characteristics, select appropriate methods from a large body of algorithms, and determine a suitable execution order. Existing methods, however, provide limited support for this decision. First, layout optimization knowledge is scattered across decades of literature and lacks unified structural organization, making decision rationales difficult to retrieve and reuse. Second, algorithm recommendations in existing tools often rely on hard-coded rules or human expertise, with little traceability to the underlying knowledge sources. Third, recommendations are typically presented as final layouts or parameter lists, which limits interactive understanding and validation. The rapid development of large language models and knowledge-enhanced reasoning offers new opportunities to overcome these limitations and to shift from experience-driven to knowledge-driven automatic graph layout optimization.

To alleviate these limitations, three challenges should be solved: (1) organizing scattered layout optimization knowledge into a structured and retrievable knowledge system; (2) automatically retrieving relevant knowledge based on graph structural characteristics and generating traceable layout operation sequences; and (3) providing clear operation explanations to support users' understanding and validation of automated decisions. We propose NOVA, a knowledge-enhanced visual analysis framework for automatic graph layout optimization. First, we construct a graph layout optimization knowledge graph covering 42 visualization papers and 3,371 knowledge triples, explicitly encoding scattered domain knowledge into a structured and retrievable knowledge base. Second, we develop a retrieval-augmented generation framework based on RAG and introduce a submodular subgraph compression strategy (SGBP-SC) to retrieve decision evidence relevant to the current

*College of Computer Science and Technology, Zhejiang University of Technology. E-mail: 202103150227@zjut.edu.cn
†Corresponding author. Zhejiang Key Laboratory of Film and TV Media Technology, School of Media Engineering, Communication University of Zhejiang. E-mail: litong@cuz.edu.cn

graph instance, guiding large language models to generate traceable layout operation sequences and natural language explanations. Finally, we design and implement an interactive visual analysis system that supports transparent reasoning by jointly presenting graph features, layout results, and knowledge evidence, enabling users to understand and validate layout optimization decisions. To the best of our knowledge, NOVA is the first visual analysis approach that applies a knowledge graph-enhanced retrieval-generation framework to automatic graph layout optimization. Our contributions are summarized as follows:

- A design space for graph layout optimization. Through formative studies and literature review, we construct a design space covering node-level, edge-level, layout-level, and visual-level operations, providing decision guidance for layout operation recommendation.
- A knowledge-driven graph layout optimization framework. We propose a knowledge graph-based and retrieval-augmented generation framework that automatically generates layout operation sequences with reasoning traceability and scenario adaptability.
- An interactive visual analysis system. We design and implement a visual analysis system for graph layout optimization, evaluate layout improvements using visual quality metrics, and validate system effectiveness and usability through comparative experiments and user studies.

2 Related Work

2.1 Graph Layout

Graph Layout Algorithms. Force-directed layout methods were first proposed by Eades [14] and further refined by Fruchterman and Reingold [17]. These methods model a graph as a physical system and iteratively converge to an energy-minimized layout state. Building on this approach, Kamada et al. [25] defined the ideal distance between nodes as the graph-theoretic shortest path length, enabling the layout to better preserve the global topological structure. As a complement to layout algorithms, edge bundling reduces visual clutter by aggregating spatially proximate edges into bundles. Holten et al. proposed Force-Directed Edge Bundling [22], which discretizes edges into segments and applies spring forces to generate smooth bundled edge flows. Gansner et al. introduced Multilevel Agglomerative Edge Bundling [18], which uses a greedy merging strategy to improve scalability. Hurter et al. proposed Kernel Density Estimation Edge Bundling [23], which attracts sampled points on edges toward local density maxima and achieves higher computational efficiency than FDEB.

Graph Preprocessing. Large-scale graph layout usually requires preprocessing to reduce the size [20]. Sampling methods approximate the original graph by retaining a subset of nodes or edges. Leskovec et al. [28] evaluated sampling strategies, including random node sampling, random edge sampling, and random-walk sampling, and found that random-walk-based methods performed better. Multilevel coarsening [46, 47] iteratively applies matching-contraction to compress the original graph into a multi-resolution hierarchy, thereby accelerating layout computation. Community detection identifies densely connected node groups to form abstract units based on graph structure. Louvain [8] greedily optimizes modularity to merge nodes into communities level by level, while Leiden [45] further introduces a refinement phase to ensure the communities are well-connected.

2.2 LLMs and RAG

Large Language Models. In recent years, large language models (LLMs) such as GPT [9], PaLM [10], and LLaMA [44] have demonstrated strong capabilities in natural language understanding and generation tasks. Instruction tuning [37] and reinforcement learning from human feedback (RLHF) further align models with human intent, giving rise to conversational models with complex instruction-following abilities, such as ChatGPT, GPT-4 [35], and Claude. In terms of reasoning, Chain-of-Thought (CoT) prompting [51] significantly improves performance on multi-step reasoning tasks by guiding models to generate intermediate reasoning steps; ReAct [55] further integrates reasoning with action, enabling models to invoke external tools. However, LLMs still face incomplete knowledge coverage and hallucination risks in specialized domains, which constitute major obstacles to their application in knowledge-intensive recommendation tasks.

Retrieval-Augmented Generation. RAG [30] combines dense passage retrieval with sequence generation in an end-to-end framework. During inference, it retrieves relevant text chunks to condition generation, reducing hallucinations and extending the model's knowledge boundary. However, RAG's passage-level independent retrieval fails to capture relational structures among concepts, limiting its performance in multi-hop reasoning or structured decision-making tasks. To incorporate relational knowledge, graph-enhanced RAG methods have recently emerged. Edge et al. [15] construct an entity knowledge graph from a corpus with pre-generated community summaries, improving answer comprehensiveness for global semantic understanding queries over large document collections. LightRAG [19] integrates graph structures with vector representations, and uses a dual-level retrieval mechanism to efficiently query fine-grained entities and high-level topics.

2.3 Interactive and Automatic Graph Layout Improvement

Interactive and Constrained Graph Layout. User-guided and constraint-based layouts allow user intent and domain constraints to be injected into the drawing process. Dwyer et al. [13] proposed a topology-preserving constrained layout that improves an initial layout while maintaining its topology. Wang et al. [50] reformulated stress majorization into a unified framework that allows users to impose and satisfy various layout constraints during interaction. Hoffswell et al.'s SetCoLa [21] further improves the reusability of constraint-based layouts through declarative high-level constraints. These methods are user-centered but typically require users to possess layout knowledge to specify constraints.

Automatic and Learning-based Layout Models. To improve the generality and quality of force-directed layouts, Xue et al.'s Taurus [54] proposes a unified force representation and a universal solver, and Zhong et al. [58] introduce a new force based on the t-distribution to improve neighborhood preservation. Dimensionality-reduction and learning-based methods further improve scalability: DRGraph [59] achieves efficient layout of large-scale graphs via a sparse distance matrix and negative sampling, while DeepDrawing [49] (based on a graph-LSTM) and DeepGD [48] (based on a graph neural network) learn the mapping from graphs to layouts from data. For multicriteria drawing, Xue et al.'s AutoFDP [53] automatically constructs a suitable force-based model according to user-specified readability criteria. These methods focus on improving and automating the layout algorithm or model itself.

LLMs for Graph Layout. With the rise of large language models, researchers have begun to explore their potential in graph drawing. Fan et al. [16] evaluate the capabilities of LLMs in graph understanding, layout generation, and layout evaluation, finding that LLMs can perform basic reasoning with code support but struggle with complex topology, and that their generated layouts are sometimes visually acceptable yet inconsistent.

Distinction from NOVA. Unlike the above works, NOVA does not propose a new layout algorithm or solver, nor does it ask the LLM to directly generate coordinates. Instead, it is a knowledge-

driven workflow that, based on a graph's structural features, retrieves evidence from literature knowledge to recommend and explain a sequence of layout operations spanning the node, edge, layout, and visual levels, and supports users in inspecting and validating these decisions through an interactive visual analysis interface.

3 Design Requirements and Design Space

3.1 Formative Study

To identify the limitations of traditional graph layout workflows, we conduct a two-hour semi-structured interview with experts. The participants include a professor specializing in visual analysis research (E1, male, 38 years old) and a PhD student with extensive experience in large-scale graph layout (E2, male, 31 years old). The interview focuses on three topics: **TP1**: typical workflows for layout optimization and knowledge dependencies; **TP2**: limitations of existing tools in layout decision-making; and **TP3**: expectations for controllability and interpretability in automated systems. Two researchers independently code the interview data and summarize core themes using the affinity diagram method [41]. Finally, we identify the following four key design requirements:

R1 Clarify the Operation Space of Graph Layout Optimization. Graph layout optimization involves multiple strategies (e.g., node sampling, filtering, backbone extraction, edge bundling), which may cause information overload and blind search. This paper needs to build a structured operational space that organizes these strategies into an ordered framework, offering clear decision boundaries for layout optimization paths.

R2 Build a Knowledge Corpus for Graph Layout Optimization. Automated layout requires knowledge-based support rather than relying solely on experience or fixed rules. This paper should build a structured knowledge corpus from accumulated methods in the visualization community. Such a corpus ensures the applicability of methods across different graph types and operation scenes.

R3 Explore Methods for Generating Layout Operation Sequences. Knowledge is abstract and large-scale, while layout optimization operations are concrete and can converge. The paper should explore methods for generating layout operation sequences based on the above knowledge corpus, enabling the determination of optimal or near-optimal sequences within the complex knowledge search space.

R4 Support Interactive Visual Analysis of Layout Strategies. Automated systems with low transparency hinder effective human-AI collaboration. This paper should provide visualizations of automated layout operations, along with interactive inspection of their knowledge rationale and visual effects, helping users understand and verify layout strategies by the human-in-the-loop process.

3.2 Design Space

Graph layout optimization is a process driven by the joint use of multiple strategies, including node sampling, edge bundling, and force-directed operations. For automated systems, the lack of clear operation dimensions and boundaries can easily lead to blind search. Therefore, we build VEPC, a design space for graph layout optimization, which organizes scattered optimization strategies into a structured guidance framework (**R1**).

Based on the key attributes of graph layout, we define a design space with four levels: node-level operations ($\mathbb{V}$), edge-level operations ($\mathbb{E}$), layout-level optimization ($\mathbb{P}$), and visual-level enhancement ($\mathbb{C}$). This design space covers the process of large-scale graph layout optimization, from structural optimization and relation organization to visual communication, as shown in Table 1.

- **Node-level** Operations act on the node set of a graph. Large-scale graphs often suffer from rendering pressure and visual aliasing. These operations control the scale and composition of the graph through node filtering, sampling, aggregation, and related methods, producing a node subset with an appropriate scale and representative structure.
- **Edge-level** Operations act on the edge set of a graph. Large-scale or dense graphs commonly suffer from visual clutter. These operations reduce the visual interference of less important edges through filtering, sampling, and related methods, thereby extracting the key topological structure of the graph.
- **Layout-level** Optimization acts on the spatial positions of nodes. The geometric routing of edges also affects graph readability. The optimization focuses on edge bundling to reshape edge paths, reduce visual crossings and occlusion, and make the structural features of the graph easier to identify.
- **Visual-level** Enhancement acts on the visual channels of elements. It does not change the graph structure or layout. Instead, it strengthens key information through differentiated mappings of color, size, opacity, and other channels, improving graph readability and cognitive efficiency.

3.3 Graph Dataset

The datasets used in this paper mainly come from three sources: open-source benchmarks commonly used in graph layout and edge bundling studies [22, 23, 42], general graph data repositories, including Network Repository [40] and SNAP [29], and real-world open data platforms, including OpenFlights [5], Divvy Bikes [3], and HSL Helsinki Region Transport [4].

Based on these sources, we select datasets along three dimensions: application domain, scale, and topological complexity. Finally, we built a dataset containing 22 real-world graph data (Table 2). By application domain, the datasets are divided into four categories: transportation and airline networks, power and electrical networks, social and interpersonal networks, and other types. The datasets vary in scale and density. Specifically, the number of nodes ranges from 43 to 188,073, and the number of edges ranges from 159 to 548,350, covering various topological structures from sparse to dense. Different data topics also show differences in average degree. For example, sparse power networks have an average degree of about 4.90, while dense social and interpersonal networks reach 22.46.

All datasets are preprocessed in a unified way. For datasets with spatial coordinates, the original geographic coordinates are retained as optional attributes.

4 NOVA Framework

4.1 Knowledge Corpus

In this section, we construct a knowledge corpus for graph layout optimization (**R2**) to form a structured and retrievable knowledge base. The construction consists of three steps: literature collection, semantic chunking, and vector indexing, which respectively handle the selection of literature sources, the segmentation of text semantics, and the encoding of vector representations.

Literature Collection. The corpus prioritizes domain relevance as the primary selection criterion, focusing on three themes: graph layout algorithms, edge bundling methods, and large-scale graph visualization. The sources cover major visualization conferences and journals such as IEEE VIS, IEEE TVCG, and EuroVis. A total of 42 papers are finally included, covering classical force-directed layouts, hierarchical edge bundling, deep learning-based graph drawing, as well as graph sampling and coarsening strategies. The collection spans both foundational works and recent advances.

Chunking and Vector Indexing. The chunking of literature corpus affects the semantic granularity and retrieval quality of search results [43]. Fixed-size chunking tends to break the semantic integrity of paragraphs, while overly large chunks introduce irrelevant content and weaken semantic matching with the query. To

Table 1: The VEPC design space for large-scale graph layout optimization, encompassing node-level operations, edge-level operations, layout-level optimization, and visual-level enhancement.

Level	Operation	Method
Node-level Operations $\mathbb{V}$	 Node Filtering	Degree-Threshold Filtering
		Isolated Node Removal
	 Node Sampling	Breadth-First Sampling
		Random-Walk Sampling
		Degree-Weighted Sampling
	 Node Aggregation	Spatial-Grid Aggregation
		Matching-Based Coarsening
	 Community Merging	Louvain-based Merging
		Leiden-based Merging
		Infomap-based Merging
	 Skeleton Extraction	Degree Top-k Extraction
		Betweenness Top-k Extraction
		PageRank Top-k Extraction
		K-Core Decomposition
Edge-level Operations $\mathbb{E}$	 Edge Filtering	Weight-Threshold Filtering
		Length-Threshold Filtering
		Betweenness-Centrality Filtering
	 Edge Sampling	Random-Walk Edge Sampling
		Forest-Based Edge Sampling
		Degree-Weighted Edge Sampling
		Weight-Weighted Edge Sampling
		Reservoir Edge Sampling
Layout-level Optimization $\mathbb{P}$	 Edge Bundling	Grid-Based Edge Bundling
		Force-Directed Edge Bundling
		Hierarchical Edge Bundling
		Kernel-Density Edge Bundling
		Skeleton-Based Edge Bundling
Visual-level Enhancement $\mathbb{C}$	 Visual Channel Update	Color Mapping
		Size Mapping
		Opacity Mapping

Table 2: Summary statistics of datasets, including category, node range, edge range, and average degree. Node and edge ranges are reported in units of 10^3, and the number in parentheses after each category denotes the number of datasets in that category.

Dataset	Node Range	Edge Range	Degree
Transportation & Aviation (7)	0.235∼188.073	0.201∼548.350	11.766
Power & Electrical (4)	0.135∼4.941	0.335∼6.594	4.897
Social (9)	0.043∼5.242	0.159∼88.234	22.462
Others (2)	1.005∼6.517	9.780∼25.571	26.944

determine the suitable chunking scheme for our paper, we conduct a FAISS-based [24] retrieval experiment on 30 sample instances extracted from the corpus. Specifically, we evaluate the retrieval performance of four chunking strategies (fixed-size, sentence-level, structure-aware, and LLM-based semantic chunking) across multiple embedding models (text-embedding-3-small [36], Qwen3-Embedding-0.6B [57], nomic-embed-text-v1.5 [34], mxbai-embed-large-v1 [27]) for vector retrieval.

Evaluation Metrics. Let the query vector be $\mathbf{q}$, and the top-k semantic chunks retrieved be ranked in descending order of similarity as $\mathscr{D} = \{d_1, d_2, \ldots, d_k\}$. The cosine similarity between a chunk $\mathbf{d}_i$ and the query is defined as $s_i = \frac{\mathbf{q}\cdot\mathbf{d}_i}{\|\mathbf{q}\|\cdot\|\mathbf{d}_i\|}$. The evaluation metrics include top-1 similarity ($sim@1$), average top-3 similarity ($sim@3$), standard deviation of top-3 similarity ($stds@3$), and hit rate at a threshold of 0.75 ($hit@0.75$). Detailed definitions are provided in the supplementary material Equations (1)–(4).

Experimental results. As shown in Table 3, for the chunking strategy dimension, Structure-Aware chunking yields the highest $sim@1$ and $sim@3$ across all models, indicating that segmenting at semantic unit boundaries helps preserve the contextual integrity needed for retrieval. Although the LLM-Based chunking method

Table 3: Comparison of chunking strategies and embedding models for knowledge corpus vectorization and retrieval.

Chunking Method	Embedding Model	$sim@1$	$sim@3$	$stds@3$	$hit@0.75$
Fixed-Size	t3s	0.6821	0.6629	0.0166	20.00%
	Qwen	0.7476	0.7245	0.0184	46.67%
	nomic	0.8061	0.7915	0.0118	93.33%
	mxbai	0.8296	0.8192	0.0089	100.00%
Sentence-Level	t3s	0.6817	0.6636	0.0148	13.33%
	Qwen	0.7343	0.7135	0.0165	46.67%
	nomic	0.8016	0.7907	0.0092	86.67%
	mxbai	0.8297	0.819	0.0087	96.67%
Structure-Aware	t3s	0.6967	0.6744	0.0177	23.33%
	Qwen	0.7524	0.7293	0.0185	50.00%
	nomic	0.8169	0.8049	0.0101	100.00%
	mxbai	0.8388	0.8271	0.0094	100.00%
LLM-Based	t3s	0.6899	0.6705	0.0162	20.00%
	Qwen	0.7368	0.7207	0.013	40.00%
	nomic	0.8101	0.7978	0.0109	93.33%
	mxbai	0.8328	0.8223	0.0086	100.00%

Table 4: Summary of papers and semantic chunks by operation level.

Operation Level	Papers	Semantic Chunks
Node-level Operations	10	343
Edge-level Operations	6	236
Layout-level Optimization	26	1271
Total	**42**	**1850**

maintains stable $stds@3$, its $hit@0.75$ is relatively low, suggesting unstable segmentation granularity: over-merging adjacent content blurs the topical focus of each chunk and reduces the discriminability of the most relevant chunk in the vector space. For the vector indexing dimension, the embedding model determines the quality of the vector representation of semantic chunks: mxbai-embed-large-v1 achieves the strongest overall performance, which may be attributed to its optimization for retrieval tasks using the AnglE loss, whereas the general-purpose embedding model text-embedding-3-small performs relatively weakly in our scenario. Based on the above, we adopt the structure-aware strategy as our corpus chunking scheme and use mxbai-embed-large-v1 to encode the semantic chunks into vectors and build the retrieval index, yielding 1.85k semantic chunks (Table 4) that primarily cover the three operational levels of the design space VEPC (Section 3.2).

4.2 Knowledge Graph-Enhanced Layout Methods

In this section, we propose a knowledge graph–enhanced RAG method (Fig. 2) for automated layout operation sequence generation (**R3**). It navigates the domain knowledge space through structured retrieval to produce optimal or near-optimal operation sequences, while ensuring the adaptability and traceability.

Knowledge Graph Construction.

Consideration. Vector retrieval alone is insufficient for high-quality decision-making in graph layout optimization. (1) Knowledge structure is hard to expose explicitly. Layout optimization requires combining multiple optimization goals, yet the relevant knowledge is fragmented across chunks, with relational information buried in redundant natural language. (2) Inter-knowledge relations are hard to identify. Different studies may corroborate, complement, or contradict each other regarding the same layout algorithm, and plain-text retrieval cannot capture these relations at a structural level. To alleviate this, we introduce a knowledge graph that explicitly encodes scattered knowledge relations as a traversable structure, enhancing RAG's multi-hop reasoning capability in graph layout optimization [56]. Specifically:

Ontology Definition. The KG ontology comprises 5 entity types (graph types, structural features, algorithms, visual dimensions, and execution conditions) and 8 relation types spanning four semantic categories: algorithm applicability, visual effects, operation ordering, and feature–type mapping (full ontology definition in supplementary material Fig. 5). For example, the triple $< fdeb, IMPROVES, clutter\_reduction >$ denotes that "*FDEB reduces visual clutter*", while $< node\_filter, PRECEDES, fdeb >$ indicates that "*node filtering should be executed prior to FDEB*".

Knowledge Extraction. Under the ontology constraints, we apply GPT-5.4-mini to perform triple extraction over the 1,850 chunks in the corpus (Section 4.1), with each triple annotated by a confidence score and a textual evidence span. The prompt design is shown in supplementary material Fig. 6. This procedure yields 3,371 valid triples. To assess the extraction quality, we invited three researchers with long-term experience in large-scale graph layout to independently annotate the correctness of 300 triples, stratified by relation type, against their textual evidence, with the final labels aggregated by majority voting. The overall accuracy is 88.7%, and all errors are concentrated in low-confidence triples (confidence ≤ 0.3), which receive low weights in the subsequent retrieval and thus have limited impact on the system. In addition, the current knowledge mainly covers the node-level, edge-level, and layout-level operations of the VEPC design space; visual-level knowledge is rarely discussed systematically in the existing literature, and the corresponding decisions are instead supported by the VLM's visual perception of the rendered layout.

KG Output. After entity-level merging of the extracted triples, we obtain a densely connected knowledge graph in which each entity participates in over a hundred triples on average ($average_{degree} = 124.85$), jointly characterized by cross-literature layout optimization knowledge.

Retrieval-Augmented Generation.

Input. The input data originates from the graph data described in Section 3.3. However, raw graph data are typically large and contain details irrelevant to layout decisions, making them unsuitable as direct input for KG retrieval. Therefore, we abstract each graph into an 11-dimensional structured feature vector following established taxonomies from graph theory [33, 11] and visualization research [12]. The vector encodes three categories of properties: scale features (node size, edge size, graph density), structural features (presence of hierarchy, directedness, number of connected components, average degree, and degree variance), and spatial features (number of edge crossings, node overlap rate, and visual clutter).

Retrieval. Given the input feature vector, we extract a compact knowledge subgraph from the KG through three steps: feature-to-node mapping, subgraph recall, and subgraph compression, which together provide structured decision evidence for layout sequence generation. The steps are as follows:

(1) Feature-to-Node Mapping. KG nodes are represented symbolically, whereas the input feature vector takes continuous numerical values. We use GPT-4o to generate a VLM visual description, and then GPT-5.4-mini to fuse it with the 11-dimensional feature vector and discretize the continuous features into symbolic KG nodes drawn from a predefined vocabulary. This mapping yields two groups of symbolic nodes: GraphType nodes capturing the semantic category of the graph, and Condition nodes capturing the applicability prerequisites the graph satisfies. Collectively, they form the seed nodes for KG retrieval.

(2) Subgraph Recall. Starting from the seed nodes, we per-

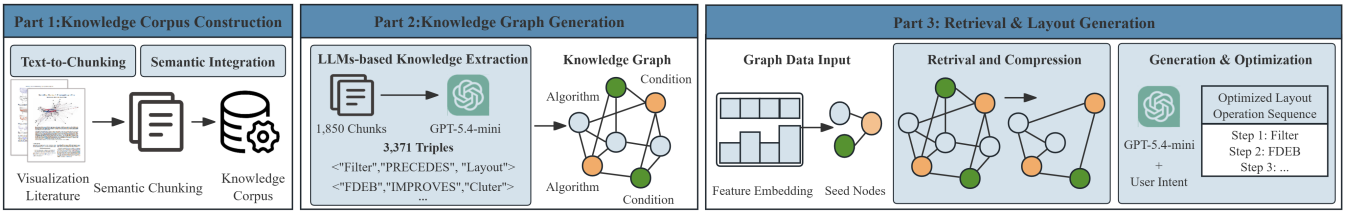

Figure 2: KG-RAG-based graph layout optimization strategy generation framework.

form breadth-first search (BFS) to recall a relevant subgraph from the KG. The traversal propagates outward along semantic relations such as algorithm applicability and execution conditions, with a depth of two hops to cover the directly associated structure of algorithm nodes while avoiding excessive noise. This yields an initial subgraph containing candidate algorithms, visual dimensions, execution conditions, and ordering constraints among algorithms.

(3) Subgraph Compression. The initial retrieval subgraph $G_0 = (V_0, E_0)$ still contains many nodes and edges weakly related to the current query, burying key reasoning paths in redundant information. Therefore, we propose SGBP-SC (Submodular Greedy with Backward Pruning, Algorithm 1).

Problem Formulation. We formulate subgraph compression as a minimum submodular cover problem: retaining all seed nodes, we seek the smallest node set satisfying the coverage constraint

$$S^* = \arg\min_S |S| \quad \text{s.t.} \quad f(S) \geq \theta \cdot f(V_0), \;\; \text{Seeds} \subseteq S \subseteq V_0,$$

where $f(\cdot)$ is a monotone submodular coverage function and θ is the coverage threshold.

Coverage Function. $f(S)$ is a weighted sum of three terms:

$$f(S) = f_{\text{app}}(S) + \mu \cdot f_{\text{imp}}(S) + \gamma \cdot f_{\text{prec}}(S),$$

where f_{app} measures the applicability of algorithm nodes to the seed nodes through the applicability weight w_{sa}, f_{imp} captures algorithms' improvement on visual dimensions, and f_{prec} rewards the execution precedence between jointly selected algorithm pairs. w_{sa} is computed from cross-paper evidence aggregation and side-effect penalization (see the supplementary material Equations (5)–(7)). All three terms are monotone submodular, so their weighted sum remains monotone submodular, which is the prerequisite for the approximation guarantee of the greedy algorithm. The parameters are determined by grid search (see the supplementary material).

Two-Phase Optimization. SGBP-SC solves the problem in two phases: a forward phase that greedily expands from the seed nodes by maximum marginal gain, adding at each step the node that maximizes the increase of $f(S)$ until $f(S) \geq \theta \cdot f(V_0)$; and a backward phase that prunes redundant nodes in ascending order of weight. Owing to the monotone submodularity of the coverage function, the forward greedy expansion enjoys the approximation guarantee $|S_{\text{greedy}}| \leq |S^*| \cdot \lceil \ln(1/(1-\theta)) \rceil$ [52], where S^* is the optimal solution. The two phases finally yield a compact knowledge subgraph G^* that serves as the structured decision evidence for layout sequence generation.

Generation. The compact subgraph is fed into GPT-5.4-mini to distill an ordered sequence of layout optimization operations, and further to select a concrete executable layout algorithm for each operation according to the graph features. NOVA then executes the sequence to produce the optimized layout, while GPT-5.4-mini generates a natural-language explanation that grounds each step in the subgraph's reasoning evidence—algorithm applicability, visual effects, and operation ordering—ensuring interpretability for users.

Algorithm 1: SGBP-SC: Submodular Greedy with Backward Pruning for Subgraph Compression

Input: Initial subgraph $G_0 = (V_0, E_0)$; seed set Seeds; weights λ, μ, γ; threshold θ
Output: Compressed subgraph G^*

foreach *algorithm–seed pair* (a, s) *with* APPLICABLE_TO *edges* **do**
  $\text{conf}^+_{sa} \leftarrow 1 - \prod_p \left(1 - \max_{\text{edges from } p} \text{conf}\right)$
  $D_a \leftarrow \frac{1}{|VD|} \sum_{vd} \text{conf}^{\text{DEG}}_{a,vd}$
  $w_{sa} \leftarrow \max(0,\ \text{conf}^+_{sa} - \lambda \cdot D_a)$
improves$[a][vd] \leftarrow$ max conf over IMPROVES edges (per (a, vd))
prec$[a_1][a_2] \leftarrow$ max conf over PRECEDES edges (per (a_1, a_2))
$f_{\text{full}} \leftarrow \texttt{f}(V_0)$
$S \leftarrow$ Seeds
while $f(S) < \theta \cdot f_{\text{full}}$ and $S \neq V_0$ **do**
  $v^* \leftarrow \arg\max_{v \notin S} \left[f(S \cup \{v\}) - f(S)\right]$
  if $v^* = \varnothing$ or gain$(v^*) \leq 0$ **then break**
  $S \leftarrow S \cup \{v^*\}$
foreach $v \in \text{sort}(S \setminus \text{Seeds})$ by weight ascending **do**
  if $f(S \setminus \{v\}) \geq \theta \cdot f_{\text{full}}$ **then** $S \leftarrow S \setminus \{v\}$
$G^* \leftarrow \texttt{Subgraph}(G_0, S)$
return G^*

5 System Design

NOVA's visual analysis interface consists of three core modules that present graph properties, layout optimization results, and knowledge evidence used in the recommendation process, respectively, supporting users in interactively analyzing the layout optimization workflow (**R4**).

The Graph Features view (Fig. 3-A) visualizes structural properties of the input graph. Continuous features are encoded as bars whose lengths are rank-normalized to indicate their relative magnitudes among all numerical features of the current graph, while Boolean features are marked with compact symbols to reduce visual redundancy.

The Action Chain view (Fig. 3-B) presents the generated layout optimization operations. The upper panel shows the LLM's reasoning based on retrieved knowledge, including its assessment of graph structural characteristics and the evidence supporting operation decisions, making algorithm selection transparent to users. The lower panel displays the operations as a sequence, with arrows indicating the chained execution of multiple layout operations.

The Graph Layout view (Fig. 3-C) renders the graph layout optimized according to the operation sequence in the Action Chain view. The top of the panel indicates the current optimization stage, while the bottom provides a step navigation bar, allowing users to jump to corresponding layout states by clicking labels and thereby review and compare the optimization process.

The KG-RAG Reasoning view (Fig. 3-D) presents the com-

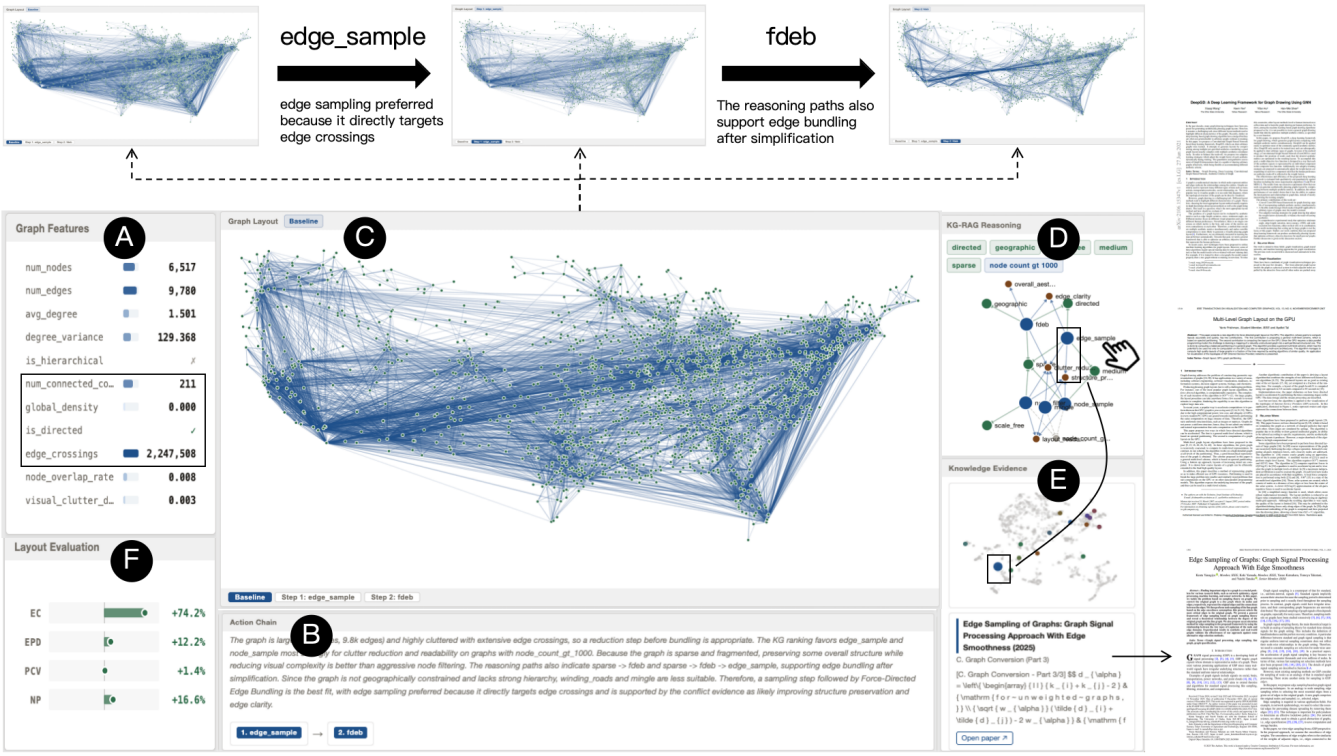

Figure 3: The NOVA visual analysis interface, demonstrated with the US Migration network. (A) Graph Features view; (B) Action Chain view; (C) Graph Layout view; (D) KG-RAG Reasoning view; (E) Knowledge Evidence view; (F) Layout Evaluation view.

pressed subgraph output by KG-RAG using a force-directed layout. Node colors encode entity types: algorithm nodes in blue, graph-type nodes in green, visual-dimension nodes in brown, and execution-condition nodes in gray. Node sizes are ordered by type (algorithm > graph type > dimension/condition), guiding attention toward core algorithm nodes.

The Knowledge Evidence view (Fig. 3-E) projects 1,850 knowledge chunks using UMAP [32]. Colors highlight semantic chunks relevant to the current reasoning process and distinguish their source papers by hue, while gray denotes the remaining corpus. Chunks from the same paper share the same color, helping users perceive the distribution of literature sources. A paper list below the view supports direct access to the original papers. When users select an associated edge in the KG-RAG Reasoning view, the corresponding knowledge chunks are enlarged and outlined, and a detail card shows the specific section and paragraph, enabling traceability from reasoning paths to original literature.

The Layout Evaluation view (Fig. 3-F) uses a dumbbell chart to show the improvement rates of four layout metrics (edge crossings *EC*, edge pixel density *EPD*, path-length coefficient of variation *PCV*, and neighborhood preservation *NP*) relative to the initial state. Dots are positioned around a vertical baseline (0%): rightward green dots indicate improvement, whereas leftward red dots indicate degradation. This design compactly conveys both the direction and magnitude of metric changes within the sidebar.

6 Case Study

We invite E3, a Ph.D. student with extensive experience in large-scale network analysis, to conduct an exploratory evaluation of NOVA on the US Migration dataset. The following case study represents one representative run of NOVA. The study consisted of three tasks: **T1** understanding how NOVA extracts graph structural features and maps them to seed nodes for knowledge retrieval; **T2** tracing the knowledge evidence behind selected operations; and **T3** evaluating the layout optimization process and its visual quality improvements.

T1: Feature Extraction and Knowledge Mapping. After E3 uploaded the US Migration dataset, the Graph Features view (Fig. 3-A) displayed an 11-dimensional structural feature vector. E3 observed that the graph had a very low average degree (1.501), 211 connected components, and near-zero global density, indicating a highly sparse directed graph. Meanwhile, the number of edge crossings reached 2,247,508, revealing severe visual clutter. The KG-RAG Reasoning view (Fig. 3-D) then presented the SGBP-SC-compressed knowledge subgraph. The label area at the top showed six seed nodes covering key characteristics of the graph in terms of scale, structure, and spatial properties. E3 noticed that algorithm nodes fdeb and edge_sample were directly connected to graph-type, visual-dimension, and execution-condition nodes, indicating their high relevance to the current graph instance in the SGBP-SC coverage.

T2: Knowledge Evidence Tracing. E3 examined the Action Chain view (Fig. 3-B) and read the LLM-generated reasoning text. NOVA identified the graph as large-scale with severe edge crossings, recommending a density-reduction step before bundling. Since the graph was sparse and highly fragmented, preserving overall structure was preferred over aggressive node filtering; thus, node_filter was excluded. As the graph lacked geographic constraints, winding_roads and mingle were less suitable, while fdeb had the strongest retrieval evidence, yielding a two-step sequence: edge_sample → fdeb. E3 then clicked the edge between edge_sample and clutter_reduction in the KG-RAG Reasoning view (Fig. 3-D), triggering the Knowledge Evidence view (Fig. 3-E) to highlight the corresponding chunk. The card showed a passage from "Edge Sampling of Graphs: Graph Signal Processing Approach With Edge Smoothness" explaining edge sampling's effect

on visual clutter reduction. E3 further clicked Open paper to verify the source, noting that tracing the reasoning path to the original publication substantially increased his trust in the recommendation.

T3: Layout Optimization Evaluation. Using the step navigation bar in the Graph Layout view (Fig. 3-C), E3 stepped through the layout states. In the initial layout, dense cross-region edges made the graph almost unreadable. After edge_sample, low-weight edges were selectively removed, reducing visual density and revealing major connection paths. After fdeb, remaining edges were bundled into coherent directional flows, clearly presenting the main migration patterns with greatly reduced local crossings. The Layout Evaluation view (Fig. 3-F) quantified these improvements: EC by 74.2%, EPD by 12.2%, PCV by 1.4%, and NP by 16.6%. All four metrics show positive gains, demonstrating the effectiveness of the knowledge-driven operation sequence on large-scale sparse directed graphs.

7 Evaluation

7.1 Comparative Experiment

In this section, we conduct comparative experiments between our method and mainstream AI programming tools to evaluate the effectiveness of our approach in graph layout optimization tasks.

Experimental Design. We select Codex (powered by GPT-5.5) and Cursor (powered by Composer 2.5) as representative baselines, both under their default settings. The experimental data consists of 15 typical graph instances from our dataset (Section 3.3), covering various types and scales. For each graph instance, all methods receive identical input: an initial graph state file in JSON format containing node coordinates and edge paths. The prompts for the AI tools explicitly specify the task objectives (e.g., reducing visual clutter and enhancing readability) and our operation space (Section 3.2), with the complete task instruction provided in the supplementary material (Fig. 12). Each method runs independently ten times per graph instance and outputs an optimized graph state file in JSON format; the table reports the average of the four metrics over the ten runs. The models and generation settings of NOVA's pipeline are detailed in the supplementary material.

Evaluation Metrics. We quantify layout performance across four dimensions widely adopted in graph layout evaluation [12], each associated with one evaluation metric: visual clarity (edge crossings, *EC*), layout balance (path-length coefficient of variation, *PCV*), spatial resources (edge pixel density, *EPD*), and topological fidelity (neighborhood preservation, *NP*). These metrics characterize complementary aspects and should be interpreted jointly rather than as individual monotonic indicators of layout quality. Each metric is measured by the improvement rate of the optimized layout relative to its initial state.

- *EC* measures the total number of intersections between non-adjacent edge pairs in G [38]. Let $\mathbf{1}[\cdot]$ be the indicator function.
$$EC = \sum_{1 \le i < j \le m} \mathbf{1}[e_i \cap e_j \neq \emptyset]$$
- *EPD* measures ink utilization [31]. It is defined as the ratio of non-background pixels to total pixels in a $W \times H$ rendering.
$$EPD = \frac{\sum_p \text{ink}(p)}{W \times H}$$
- *PCV* reflects the relative dispersion of rendered edge lengths [39].
$$\text{PCV} = \frac{\sigma(L)}{\mu(L)}$$
- *NP* defines the average overlap rate between the topological k-neighbors and the spatial k-neighbors of all nodes in G [26].
$$NP = \frac{1}{N} \sum_{i=1}^{N} \frac{|\mathscr{T}_k(i) \cap \mathscr{S}_k(i)|}{k}$$

Experimental results. The statistical results are shown in Table 5. NOVA achieves the best performance in visual clarity *EC* and spatial resource utilization *EPD*, with average improvements of +23.6% and +10.3%, respectively. In contrast, Codex and Cursor both degrade *EPD* by -25.7% and -14.4%. For layout balance *PCV*, the three methods show relatively small differences: NOVA (+1.3%) is close to Cursor (+1.7%), whereas Codex exhibits a clear degradation (-9.1%). For topological fidelity *NP*, NOVA (+1.0%) underperforms Codex (+19.8%) and Cursor (+9.5%).

NOVA's advantage is most pronounced in visual clarity. It improves *EC* on most datasets, with particularly large gains on Dolphin Social (+86.4%) and US Power Grid (+86.7%). By contrast, Cursor achieves only a +2.8% average *EC* improvement, suggesting that general-purpose AI coding tools struggle to generate graph-aware edge-crossing reduction strategies.

The *EPD* results reveal a common limitation of baseline methods. Negative *EPD* values indicate excessive removal of edges or nodes, producing overly sparse layouts. Codex degrades *EPD* by -115.7% on JAZZ Collaboration and -130.4% on Divvy OD, indicating a tendency to trade information preservation for visual neatness through aggressive sparsification. Although NOVA also decreases *EPD* on these datasets (-44.7% and -23.9%), the degradation is substantially smaller, showing that knowledge constraints help suppress extreme operations.

NOVA's relative weakness on *NP* reflects an inherent trade-off in layout optimization. Its average *NP* improvement is +1.0%, lower than Codex (+19.8%) and Cursor (+9.5%). Since *NP* measures the preservation of topological neighborhoods in the 2D layout, operations such as node filtering, aggregation, and edge bundling may alter graph connectivity or node positions while improving *EC* and *EPD*. Thus, NOVA's substantial gains in *EC* and *EPD* come with a relative cost in *NP*, highlighting the trade-off between visual quality and topological fidelity in graph layout optimization.

7.2 Ablation Study

In this section, we conduct ablation experiments to quantify the contribution of each knowledge component to the layout optimization performance of NOVA.

Experimental Design. We ablate the knowledge sources of the recommendation generation stage and construct two variants: (1) **w/o KG**, which removes the structured retrieval over the knowledge graph and instead builds the knowledge context solely through vector retrieval, degenerating into a naive RAG; (2) **LLM-only**, which removes all retrieval augmentation and instead generates operation sequences directly from the graph feature description and the operation space. All remaining components of both variants are identical to the full system. The experimental settings follow the comparative experiment (Section 7.1): on each of the 15 datasets, every variant is independently run 10 times and averaged, and we report the average improvement rates of the four metrics relative to the initial layout.

Experimental results. The statistical results are shown in Table 6. The full system achieves positive improvements on all four metrics and performs best on visual clarity (*EC*) and spatial resources (*EPD*), with average improvements of +23.6% and +10.3%, respectively. As the knowledge components are progressively removed, both metrics degrade monotonically: *EC* drops to +19.6% (w/o KG) and further to +6.9% (LLM-only), while *EPD* falls to +0.7% and further deteriorates to -9.3%. This gradient degradation, aligned with the completeness of the knowledge, indicates that NOVA's performance advantage stems from the intro-

Table 5: Statistical analysis of comparative experiments between NOVA, Codex and Cursor.

Dataset	Codex (GPT-5.5)				Cursor (Composer 2.5)				NOVA (ours)			
	EC	*EPD*	*PCV*	*NP*	*EC*	*EPD*	*PCV*	*NP*	*EC*	*EPD*	*PCV*	*NP*
Powernet_1[1]	+11.4%	-1.7%	+2.3%	-0.1%	-2.7%	-15.5%	+37.1%	-4.0%	-2.3%	+17.0%	+0.2%	+0.0%
Powernet_2[1]	+8.1%	-0.3%	+2.5%	+1.9%	+3.0%	-10.7%	-15.7%	+0.2%	+8.4%	-3.7%	+1.0%	+1.8%
Powernet_3[1]	+11.7%	+0.4%	+1.8%	-2.9%	+0.9%	-17.4%	+2.4%	-5.6%	+13.5%	+15.7%	+1.8%	+2.9%
US Power Grid[2]	+34.1%	-70.3%	-72.3%	+2.0%	-52.9%	+1.4%	+0.0%	-1.1%	+86.7%	+15.4%	+9.9%	-30.1%
GitHub_Global[1]	-2.4%	-18.2%	+0.1%	+0.0%	+2.7%	-18.4%	+0.0%	+0.0%	-0.1%	+36.9%	+5.5%	+0.0%
GitHub_Europe[1]	+0.5%	-15.9%	+0.0%	+0.0%	+4.2%	-13.5%	-0.5%	+1.5%	-1.3%	+53.4%	+12.0%	+0.0%
GitHub_West_1[1]	+6.5%	+4.8%	-1.2%	+29.5%	+11.6%	-11.0%	+1.4%	+35.2%	+1.6%	+0.0%	+0.0%	+0.0%
GitHub_West_2[1]	+15.3%	-18.2%	+1.0%	-0.9%	+2.7%	-28.0%	-0.0%	+0.0%	+75.0%	+24.5%	+5.3%	+1.7%
Lesmis Network[2]	+14.2%	-1.7%	-24.0%	+6.0%	+15.4%	-11.8%	-4.9%	+6.0%	+6.6%	-32.3%	-2.8%	-0.2%
Dolphin Social[2]	+6.3%	-19.9%	-59.4%	-1.1%	+6.3%	-16.2%	-2.4%	-1.1%	+86.4%	+59.0%	-17.6%	+37.8%
JAZZ Collaboration[2]	+11.8%	-115.7%	-27.6%	-5.6%	-0.7%	-15.9%	+0.0%	-4.2%	+5.0%	-44.7%	+1.0%	+0.7%
US Migration[7]	+73.2%	-3.8%	+10.9%	+166.8%	+46.4%	-9.8%	+1.6%	+42.7%	+2.7%	+4.7%	+0.3%	+0.0%
US Airline[6]	+9.8%	-22.8%	+5.2%	-13.1%	-3.5%	-39.5%	-0.8%	-3.6%	+74.2%	+24.1%	+2.4%	-2.8%
US Airline Variant[1]	+35.0%	+27.6%	+10.3%	+150.4%	-0.1%	+0.3%	+0.0%	+102.6%	-0.3%	+7.7%	+0.3%	+0.0%
Divvy OD[3]	+26.9%	-130.4%	+14.0%	-36.3%	+8.9%	-9.9%	+7.6%	-26.6%	-2.8%	-23.9%	+0.7%	+2.7%
Average	**+17.5%**	**-25.7%**	**-9.1%**	**+19.8%**	**+2.8%**	**-14.4%**	**+1.7%**	**+9.5%**	**+23.6%**	**+10.3%**	**+1.3%**	**+1.0%**

Table 6: Statistical analysis of ablation experiments between NOVA, w/o KG and LLM-only.

Variant	*EC*	*EPD*	*PCV*	*NP*
NOVA	+23.6%	+10.3%	+1.3%	+1.0%
w/o KG (naive RAG)	+19.6%	+0.7%	+1.7%	+2.8%
LLM-only	+6.9%	-9.3%	+0.9%	+6.4%

duced knowledge components rather than the general capability of the underlying LLM.

The two variants remove the structured organization of knowledge and the entire retrieval augmentation mechanism, respectively, so the performance gaps among the three configurations allow us to attribute the contribution of each component. Comparing LLM-only with w/o KG, introducing retrieval augmentation substantially restores both *EC* and *EPD* (*EC*: +6.9% $\rightarrow$ +19.6%; *EPD*: -9.3% $\rightarrow$ +0.7%), indicating that the knowledge retrieved from the literature corpus accounts for the major part of the optimization gains. On this basis, the marginal contribution of the knowledge graph is concentrated on *EPD* (+0.7% $\rightarrow$ +10.3%): graph layout optimization knowledge is scattered and implicit across discrete text chunks, and the knowledge graph encodes it explicitly into a traversable structure that supports operation selection and ordering through multi-hop reasoning. In addition, the *EPD* degradation of LLM-only is consistent with the failure mode of Codex and Cursor in the comparative experiment (*EPD* of -9.3%, -25.7%, and -14.4%, respectively), further corroborating that the retrieved knowledge suppresses the tendency toward aggressive sparsification.

For layout balance *PCV*, the three configurations show relatively small differences. For topological fidelity *NP*, the metric rises slightly as the knowledge components are removed (+1.0% $\rightarrow$ +2.8% $\rightarrow$ +6.4%); however, across all methods in the comparative and ablation experiments, *NP* consistently varies inversely with *EPD*—the increase stems from conservative optimization or over-sparsification, reflecting the trade-off between visual quality and topological fidelity discussed in Section 7.1 rather than better layout quality.

7.3 User Study

In this section, we collect participant feedback on NOVA through the user study, with a focus on visual design and usage experience.

Setup. We recruit 29 participants (P1-P29, 17 male, 12 female, 13 undergraduates, 16 graduate students, $age_{mean} = 25.1$). As shown in Fig. 4, the scatter plot represents participants (symbols denote specific research domains), with their positions in the orthogonal quadrants reflecting research experience and interest in graph layout. Overall, the recruited participants have backgrounds closely aligned with our work, primarily from data visualization, with several focusing specifically on graph visualization. We conduct the evaluation through in-person sessions with a questionnaire. We first spend 15 minutes introducing the functionality and workflow of NOVA. Participants then complete three exploration tasks: (1) understanding the layout operations recommended by NOVA and the rationale behind them; (2) tracing the knowledge evidence underlying a recommended operation to judge its credibility; (3) comparing the layout quality before and after optimization to decide whether the recommendation is worth adopting. Building on this, participants freely explore the system, and we collect their feedback through the questionnaire.

Questionnaire Design. The questionnaire (Table 7) focuses on the graph visualization experience (Q1-Q5), visual design and usage experience (Q6-Q12), and qualitative feedback (Q13-Q14) of NOVA. It gathers expert insights into the effectiveness and novelty of the primary components. It also collects feedback on the usage experience, including the learning curve, interaction fluency, and overall satisfaction to identify potential aspects for refinement. Participants also provide qualitative feedback on the system's strengths, limitations, and desired functionalities for future iterations. The questionnaire results are quantitatively scored using the 7-point Likert scale.

Questionnaire Result. As shown in Fig. 4, participants evaluated NOVA favorably regarding its functionality and practicality. In terms of effectiveness, NOVA achieved a high score of 6.28, and 89.65% of participants found it more helpful for decision-making than traditional layout tools. Notably, the KG-RAG Reasoning View achieved a prominent score of 6.13. In addition, all participants reported high satisfaction with the overall interaction fluency while on average 81.03% of participants perceived the learning cost of the core views as low. A significant majority indicated they

Table 7: Questionnaire design in the user study.

Part	Questionnaire Content
Graph Visualization Experience	Q1 Familiarity with graph layout algorithms Q2 Familiarity with RAG techniques Q3 How do you usually select algorithms for graph layout optimization? Q4 How would you improve an unsatisfactory graph layout? Q5 What auxiliary information do you expect a graph layout tool to provide?
Visual Design and User Experience	Q6 Perceived learning cost of NOVA Q7 Usefulness of the KG-RAG Reasoning view for understanding layout decisions Q8 Usefulness of the Knowledge Evidence view for supporting layout decisions Q9 Perceived positivity of knowledge validation for layout decisions using NOVA Q10–Q11 Novelty and effectiveness of NOVA Q12 Willingness to recommend NOVA to others
Subjective Feedback	Q13 Please describe the strengths of NOVA. Q14 Please describe the limitations of NOVA and provide suggestions for improvement.

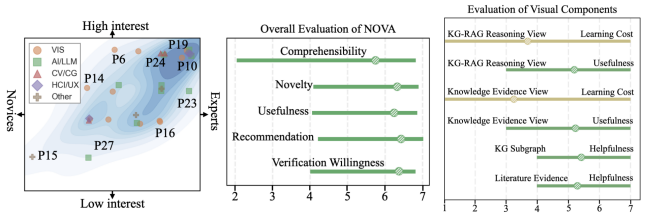

Figure 4: Participant background and selected questionnaire results.

would highly recommend the system to their colleagues. Most Participants reached a consensus that NOVA transforms the evaluation from a "black-box result" into a "white-box process". Specifically, P2 and P27 highlighted that it alleviates the confusion in layout optimization algorithms selection.

Despite the overall positive reception, participants identified several limitations and offered constructive feedback. First, several participants (P1, P11, P13) suggested incorporating richer legends and hover-based explanations to further reduce cognitive load. Second, some participants (P8, P23) noted noticeable latency during interactions, particularly when processing large-scale graph data. Furthermore, most participants advocated for a Human-in-the-loop mechanism to establish the closed feedback loop for layout decisions. Additionally, they called for features such as historical decision review and batch layout processing for multiple graph inputs.

8 Conclusion

This paper presents NOVA, a knowledge-enhanced visual analytics framework for automatic graph layout optimization. We construct a layout optimization knowledge graph, propose a submodular subgraph compression and retrieval strategy (SGBP-SC), and use it to guide large language models in generating traceable layout operation sequences. We further design and implement an interactive visual analysis system that supports transparent reasoning. Comparative experiments and user studies show that NOVA improves the traditionally experience-driven graph layout optimization workflow. Future work will expand the coverage of the knowledge corpus and incorporate human-in-the-loop mechanisms to support interactive refinement of layout results.

Acknowledgments

National Natural Science Foundation of China (62422607, 62372411, 62432014), Zhejiang Provincial Natural Science Foundation of China(QKWL25F0301)

References

[1] Divided edge bundling example datasets. https://github.com/davidselassie/DividedEdgeBundling, 2011. 9
[2] The network data repository. https://networkrepository.com, 2015. 9
[3] Divvy bikes system data. https://divvybikes.com/system-data, 2026. 3, 9
[4] HSL helsinki region transport. https://hri.fi/data/en_GB/dataset/helsingin-ja-espoon-kaupunkipyorilla-ajatut-matka, 2026. 3
[5] OpenFlights. https://openflights.org/data.php, 2026. 3
[6] US Airline dataset, d3.ForceBundle. https://github.com/upphiminn/d3.ForceBundle, 2026. 9
[7] US Migration dataset, GPU Edge Bundling. https://github.com/CreativeCodingLab/GPUEdgeBundling, 2026. 9
[8] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. 2
[9] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, et al. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 1877–1901, 2020. 2
[10] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. 2
[11] L. d. F. Costa, F. A. Rodrigues, G. Travieso, and P. R. Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, 2007. 5
[12] S. Di Bartolomeo, T. Crnovrsanin, D. Saffo, E. Puerta, C. Wilson, and C. Dunne. Evaluating graph layout algorithms: A systematic review of methods and best practices. *Computer Graphics Forum*, 43(6):e15073, 2024. 5, 8
[13] T. Dwyer, K. Marriott, and M. Wybrow. Topology preserving constrained graph layout. In *Proceedings of the International Symposium on Graph Drawing*, pp. 230–241, 2008. 2
[14] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42(1):149–160, 1984. 1, 2
[15] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024. 2
[16] Y. Fan, X. Lyu, L. Wang, Y. Zhao, F. Zhou, and Y. Wang. How well will llms perform for graph layout tasks? *Visual Informatics*, 10(1):100285, 2025. 2
[17] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. 1, 2
[18] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *Proceedings of the 2011 IEEE Pacific Visualization Symposium*, pp. 187–194, 2011. 1, 2
[19] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*, 2024. 2
[20] M. Hashemi, S. Gong, J. Ni, W. Fan, B. A. Prakash, and W. Jin. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 8058–8066, 2024. 2
[21] J. Hoffswell, A. Borning, and J. Heer. Setcola: High-level constraints for graph layout. *Computer Graphics Forum*, 37(3):537–548, 2018. 2
[22] D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):983–990, 2009. 1, 2, 3
[23] C. Hurter, O. Ersoy, and A. Telea. Graph bundling by kernel density estimation. *Computer Graphics Forum*, 31(3):865–874, 2012. 1, 2, 3

[24] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019. 4
[25] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. 2
[26] J. A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7-9):1431–1443, 2009. 8
[27] S. Lee, A. Shakir, D. Koenig, and J. Lipp. Open source strikes bread - new fluffy embedding model. https://www.mixedbread.com/blog/mxbai-embed-large-v1, 2024. 4
[28] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 631–636, 2006. 1, 2
[29] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007. 3
[30] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 9459–9474, 2020. 2
[31] A. Lhuillier, C. Hurter, and A. Telea. State of the art in edge and trail bundling techniques. *Computer Graphics Forum*, 36(3):619–645, 2017. 8
[32] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 7
[33] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003. 1, 5
[34] Z. Nussbaum, J. X. Morris, B. Duderstadt, and A. Mulyar. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024. 4
[35] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 2
[36] OpenAI. New embedding models and API updates. https://openai.com/index/new-embedding-models-and-api-updates/, 2024. 4
[37] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 27730–27744, 2022. 2
[38] H. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the International Symposium on Graph Drawing*, pp. 248–261, 1997. 8
[39] H. C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002. 8
[40] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015. 3
[41] R. Scupin. The kj method: A technique for analyzing data derived from japanese ethnology. *Human Organization*, 56(2):233–237, 1997. 3
[42] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2354–2363, 2011. 3
[43] M. A. Shaukat, M. Adnan, and C. C. N. Kuhn. A systematic investigation of document chunking strategies and embedding sensitivity. *arXiv preprint arXiv:2603.06976*, 2026. 3
[44] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2
[45] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: Guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019. 2
[46] A. D. B. Valejo, R. Fabbri, A. de Andrade Lopes, L. Zhao, and M. C. F. de Oliveira. Multilevel coarsening for interactive visualization of large bipartite networks. *Frontiers in Research Metrics and Analytics*, 7(1):855165, 2022. 1, 2
[47] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *Proceedings of the 8th International Symposium on Graph Drawing*, pp. 171–182, 2000. 2
[48] X. Wang, K. Yen, Y. Hu, and H.-W. Shen. Deepgd: A deep learning framework for graph drawing using gnn. *IEEE computer graphics and applications*, 41(5):32–44, 2021. 2
[49] Y. Wang, Z. Jin, Q. Wang, W. Cui, T. Ma, and H. Qu. Deepdrawing: A deep learning approach to graph drawing. *IEEE transactions on visualization and computer graphics*, 26(1):676–686, 2019. 2
[50] Y. Wang, Y. Wang, Y. Sun, L. Zhu, K. Lu, C.-W. Fu, M. Sedlmair, O. Deussen, and B. Chen. Revisiting stress majorization as a unified framework for interactive constrained graph visualization. *IEEE transactions on visualization and computer graphics*, 24(1):489–499, 2017. 2
[51] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 24824–24837, 2022. 2
[52] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982. 6
[53] M. Xue, Y. Wang, Z. Wang, L. Zhu, L. Cui, Y. Chen, Z. Ding, O. Deussen, and Y. Wang. Autofdp: Automatic force-based model selection for multicriteria graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 32(2):1554–1568, 2025. 2
[54] M. Xue, Z. Wang, F. Zhong, Y. Wang, M. Xu, O. Deussen, and Y. Wang. Taurus: towards a unified force representation and universal solver for graph layout. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):886–895, 2022. 2
[55] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*, pp. 1–34, 2023. 2
[56] H. Yuan, Q. Sun, J. Shi, M. Liu, J. Yuan, Z. Zhang, X. Fu, and J. Li. Retrieving minimal and sufficient reasoning subgraphs with graph foundation models for path-aware graphrag. *arXiv preprint arXiv:2603.07179*, 2026. 5
[57] Y. Zhang, M. Li, D. Long, X. Zhang, H. Lin, B. Yang, P. Xie, A. Yang, D. Liu, J. Lin, F. Huang, and J. Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. 4
[58] F. Zhong, M. Xue, J. Zhang, F. Zhang, R. Ban, O. Deussen, and Y. Wang. Force-directed graph layouts revisited: a new force based on the t-distribution. *IEEE Transactions on Visualization and Computer Graphics*, 30(7):3650–3663, 2023. 2
[59] M. Zhu, W. Chen, Y. Hu, Y. Hou, L. Liu, and K. Zhang. Drgraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1666–1676, 2020. 2

SUPPLEMENTAL MATERIALS

- $sim@1$ measures the similarity of the top-1 retrieved chunk, reflecting retrieval precision.

$$sim@1 = s_1 \tag{1}$$

- $sim@3$ computes the average similarity of the top-3 retrieved chunks, reflecting their overall quality.

$$sim@3 = \frac{1}{3}\sum_{i=1}^{3} s_i \tag{2}$$

- $stds@3$ quantifies the standard deviation of similarities among the top-3 chunks, reflecting retrieval stability.

$$stds@3 = \sqrt{\frac{1}{3}\sum_{i=1}^{3}(s_i - sim@3)^2} \tag{3}$$

Entity Definition

Entity Type	Values
GraphType	hierarchical, dense, sparse, scale_free, small_world, geographic, bipartite, community, planar, temporal, weighted, directed, large, medium, small
Feature	num_nodes,num_edges,global_density,avg_degree,degree_variance,num_connected_components,is_directed,is_hierarchical,edge_crossings,node_overlap_rate,visual_clutter_degree
Algorithm	node_filter,node_sample,edge_filter,edge_sample,fdeb,kdeeb,mingle,winding_roads,force_directed,hierarchical_edge_bundling,multilevel_layout,stress_majorization,spectral_layout
VisualDimension	edge_clarity,clutter_reduction,layout_readability,structure_preservation,overall_aesthetics
Condition	node_count_gt_1000,node_count_gt_10000,edge_density_gt_0.1,has_hierarchical_structure,has_geographic_positions,has_edge_weights,has_community_structure,has_hub_nodes,is_planar,is_directed,requires_fixed_positions

Relation Definition

Relation	Source → Target	Definition
APPLICABLE_TO	Algorithm → GraphType	Algorithm fits the graph type
REQUIRES_CONDITION	Algorithm → Condition	Algorithm requires the condition
IMPROVES	Algorithm → VisualDimension	Algorithm enhances the dimension
DEGRADES	Algorithm → VisualDimension	Algorithm harms the dimension
PRECEDES	Algorithm → Algorithm	Algorithm should run before another
HAS_FEATURE	GraphType → Feature	Graph type has the feature
INDICATES	Feature → GraphType	Feature implies the graph type
DERIVED_FROM	Condition → Feature	Condition is derived from the feature

Figure 5: Ontology Definition for Knowledge Graph Construction

Knowledge Graph Triplet Extraction Prompt

Task Description:
You are a knowledge graph extraction expert in graph visualization and layout optimization. Your task is to extract factual knowledge triplets from the given paper title and paper segment. Each triplet must follow the predefined ontology schema and must be explicitly supported by the original text.

Instructions:
- Extract only triplets that are clearly supported by the source text.
- Do not infer relationships that are not stated or strongly evidenced in the text.
- The confidence score should reflect the textual support for the triplet and range from 0.0 to 1.0.
- If no valid triplet can be extracted, return an empty JSON array [].
- Each triplet must include source node, source type, relation, target node, target type, confidence, and textual evidence.

Examples:

```
[
  {
    "src": "edge_bundling",
    "src_type": "Algorithm",
    "rel": "IMPROVES",
    "dst": "clutter_reduction",
    "dst_type": "VisualDimension",
    "confidence": 0.9,
    "evidence": "FDEB groups geometrically similar edges into smooth bundles, which substantially reduces visual clutter in dense graphs."
  },
  {
    "src": "edge_bundling",
    "src_type": "Algorithm",
    "rel": "APPLICABLE_TO",
    "dst": "geographic",
    "dst_type": "GraphType",
    "confidence": 0.9,
    "evidence": "FDEB achieves the best clutter reduction results on geographic flow maps."
  }
]
```

Input:
- **Paper Title**: textual title of the paper
- **Paper Chunk**: a text block extracted from a paper

Output Format:
Only output a **JSON** array without any additional text:

```
[
  {
    "src": "node name",
    "src_type": "node type",
    "rel": "relation type",
    "dst": "node name",
    "dst_type": "node type",
    "confidence": 0.0-1.0,
    "evidence": "a sentence or phrase from the original text"
  },
  ...
]
```

Figure 6: Knowledge Graph Triplet Extraction Prompt

- *hit*@0.75 calculates the proportion of top-3 chunks with similarity above 0.75, reflecting high-relevance hit rate.

$$hit@0.75 = \frac{1}{3}\sum_{i=1}^{3}\mathbf{1}\left[s_i \geq 0.75\right] \tag{4}$$

The formulas above define per-query metric values, while the results reported in Table 3 are the averages of each metric over the $N = 30$ test queries.

Applicability Weight w_{sa}. w_{sa} measures the strength of applicability of an algorithm node a to a seed node s, integrating cross-paper evidence aggregation and side-effect penalization. In the KG, the same (s,a) pair may have APPLICABLE_TO edges from multiple papers. Since multiple edges from the same paper are not independent evidence, we adopt a two-step Noisy-OR: we first take the maximum confidence within each paper, and then aggregate the per-paper representative confidences across papers:

$$\text{conf}^{+}_{sa} = 1 - \prod_{p}\left(1 - \max_{\text{edges from } p}\text{conf}\right). \tag{5}$$

If algorithm a has DEGRADES relations to visual dimensions, we

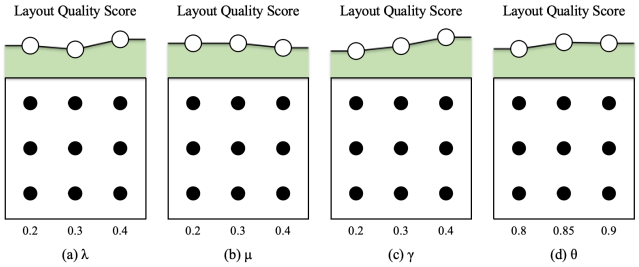

Figure 7: Layout Quality Score versus each parameter value in the grid search: (a) λ, (b) μ, (c) γ, (d) θ.

Visual Perception Prompt

Task Description:
You are a graph visualization analyst. Your task is to describe the visual characteristics of a rendered graph visualization in 2-3 sentences, focusing on the overall layout pattern, edge density and crossing situation, node distribution uniformity, and visible community or hub-and-spoke structure. The description will be used as a retrieval query for graph layout algorithm recommendation.

Instructions:
- Focus only on salient visual features of the rendered image.
- Output plain English only, no bullet points or headers.
- Keep the description within 2-3 sentences.

Examples:
"The graph exhibits a geographically distributed layout with nodes spread across the plane. Edge density is extremely high, with massive crossings between long-range edges obscuring the underlying structure. No clear community structure is visible due to severe visual clutter."

Input:
- **Chart**: a rendered graph image (PNG)

Output Format:
Plain English text (2-3 sentences).

Figure 8: Visual Perception Prompt

use the mean confidence of these edges as the penalty strength:

$$D_a = \frac{1}{|VD|}\sum_{vd}\text{conf}^{\text{DEG}}_{a,vd}. \tag{6}$$

The combined weight is

$$w_{sa} = \max\left(0,\ \text{conf}^{+}_{sa} - \lambda \cdot D_a\right), \tag{7}$$

where the non-negative truncation preserves the monotonicity of $f(S)$.

Parameter Selection via Grid Search. We determine the four parameters of SGBP-SC via grid search. The search space is $\lambda, \mu, \gamma \in \{0.2, 0.3, 0.4\}$ and $\theta \in \{0.8, 0.85, 0.9\}$, giving 81 configurations; each is run on the 10 datasets, using the equal-weighted composite of the four layout-quality improvement metrics (EC, EPD, NP, PCV) as the objective. We adopt the configuration with the highest composite score: $\lambda = 0.4$, $\mu = 0.3$, $\gamma = 0.3$, $\theta = 0.85$.

Generation Settings. The NOVA pipeline involves four LLM calls. Table 8 lists the model and temperature used in each stage. The complete prompts of the four stages are shown in Figs. 8–11.

Table 8: LLM settings for each stage of the NOVA pipeline.

Stage	Model	Temperature
Visual perception	GPT-4o	0.3
Seed-node extraction	GPT-5.4-mini	0.3
Operation sequence generation	GPT-5.4-mini	0.3
Layout algorithm selection	GPT-5.4-mini	0.1

Seed-Node Extraction Prompt

Task Description:
You are an expert in graph visualization. Your task is to fuse the graph structural features with the visual perception description, and to discretize the continuous features into two groups of symbolic seed nodes defined in the knowledge graph: GraphType nodes capturing the semantic category of the graph, and Condition nodes capturing the applicability prerequisites the graph satisfies.

Instructions:
- Only include labels that clearly apply based on the features.
- Do NOT invent labels outside the predefined vocabularies:
  GraphType: hierarchical, dense, sparse, scale_free, small_world, geographic, bipartite, community, planar, temporal, weighted, directed, large, medium, small
  Condition: node_count_gt_1000, node_count_gt_10000, edge_density_gt_0.1, has_hierarchical_structure, has_geographic_positions, has_edge_weights, has_community_structure, has_hub_nodes, is_planar, is_directed, requires_fixed_positions
- Output valid JSON only, no extra text.

Examples:
{
 "description": "This is a medium-scale, highly sparse directed graph with 6,517 nodes and 9,780 edges. The graph is fragmented into 211 connected components with near-zero global density, while edge crossings are severe, indicating high visual complexity.",
 "graph_types": ["directed", "geographic", "scale_free", "medium", "sparse"],
 "conditions": ["node_count_gt_1000"]
}

Input:
- **Graph Features**: the 11-dimensional structural feature vector
- **Visual Perception**: textual description of the rendered graph

Output Format:
Only output a **JSON** object without any additional text:
{
 "description": "a 2-4 sentence academic description of the graph",
 "graph_types": ["label1", "label2", ...],
 "conditions": ["label1", ...]
}

Figure 9: Seed-Node Extraction Prompt

Operation Sequence Generation Prompt

Task Description:
You are an expert in graph layout visualization optimization. Based on the provided graph type description and knowledge graph reasoning results, your task is to generate an ordered layout optimization action chain with explicit reasoning.

Instructions:
- Available operations: node_filter, node_sample, node_aggregation, community_merging, skeleton_extraction, edge_filter, edge_sample, edge_bundling, visual_channel_update
- Generate an ordered action chain adapted to the graph characteristics.
- Ground the reasoning in the knowledge graph evidence.
- Output valid JSON only, no extra text.

Examples:
{
 "reasoning": "The graph is sparse with severe edge crossings. A density-reduction step is applied before bundling; since the graph is fragmented, edge sampling is preferred over aggressive node filtering, followed by edge bundling for clutter reduction.",
 "action_chain": ["edge_sample", "edge_bundling"]
}

Input:
- **Graph Description**: textual description of the graph type
- **KG Reasoning Results**: the compressed knowledge subgraph from SGBP-SC

Output Format:
Only output a **JSON** object without any additional text:
{
 "reasoning": "analysis based on graph features and KG reasoning evidence",
 "action_chain": ["operation1", "operation2", ...]
}

Figure 10: Operation Sequence Generation Prompt

Layout Algorithm Selection Prompt

Task Description:
You are a graph algorithm selection assistant. For each operation in the recommended action chain, your task is to select the most suitable executable method from its candidate set in the VEPC design space, based on the graph features and graph description.

Instructions:
- Select exactly one method for each operation from its candidate set.
- Base the selection on graph features, e.g., prefer weight-based methods for weighted graphs, spatial methods when coordinates are available, and connectivity-preserving methods for fragmented graphs.
- Output valid JSON only, no extra text.

Examples:
{
 "edge_sample": "degree_weighted",
 "edge_bundling": "force_directed"
}

Input:
- **Graph Features**: node_count, edge_count, is_directed, has_weights, has_coordinates, graph_types
- **Graph Description**: textual description of the graph type
- **Candidate Methods per operation (VEPC design space)**:
  node_filter: {degree_threshold, isolated_removal};node_sample: {bfs, random_walk, degree_weighted};
  node_aggregation: {spatial_grid, matching};community_merging: {louvain, leiden, infomap};
  skeleton_extraction: {degree, betweenness, pagerank, kcore};edge_filter: {weight, length, betweenness};
  edge_sample: {random_walk, forest, degree_weighted, weight_weighted, reservoir};
  edge_bundling: {grid_based, force_directed, hierarchical, kernel_density, skeleton_based};
  visual_channel_update: {color, size, opacity}

Output Format:
Only output a **JSON** object without any additional text:
{ "operation": "method", ... }

Figure 11: Layout Algorithm Selection Prompt

Baseline Task Instruction

Task Description:
You are an expert in graph visualization optimization. Given a graph dataset and its initial layout, your task is to optimize the visualization by selecting and applying appropriate operations to the graph structure and the edge rendering, so as to reduce visual clutter and enhance readability while preserving the essential structure of the graph.

Instructions:
- Select a suitable combination of operations from the following operation space; the combination and order are at your discretion:
  node_filter {degree_threshold, isolated_removal}; node_sample {bfs, random_walk, degree_weighted};
  node_aggregation {spatial_grid, matching}; community_merging {louvain, leiden, infomap};
  skeleton_extraction {degree, betweenness, pagerank, kcore};edge_filter {weight, length, betweenness};
  edge_sample {random_walk, forest, degree_weighted, weight_weighted, reservoir};
  edge_bundling {grid_based, force_directed, hierarchical, kernel_density, skeleton_based};
  visual_channel_update {color, size, opacity}

Examples:
{
 "nodes": { "node_id_1": [x1, y1], "node_id_2": [x2, y2] },
 "edges": [
  { "source": "node_id_1", "target": "node_id_2",
   "path": [[x1, y1], [x2, y2], [x3, y3]]}
 ]
}

Input:
- **initial_positions.json**: initial graph state containing node coordinates and edge paths

Output Format:
Only output an optimized graph state file (**optimized_output.json**):
- "nodes": retained nodes with their coordinates
- "edges": retained edges with rendering paths

Figure 12: Task instruction provided to the baseline tools in the comparative experiment.