Minimizing K in K-Plane Graph Drawings

Heilbronn 43

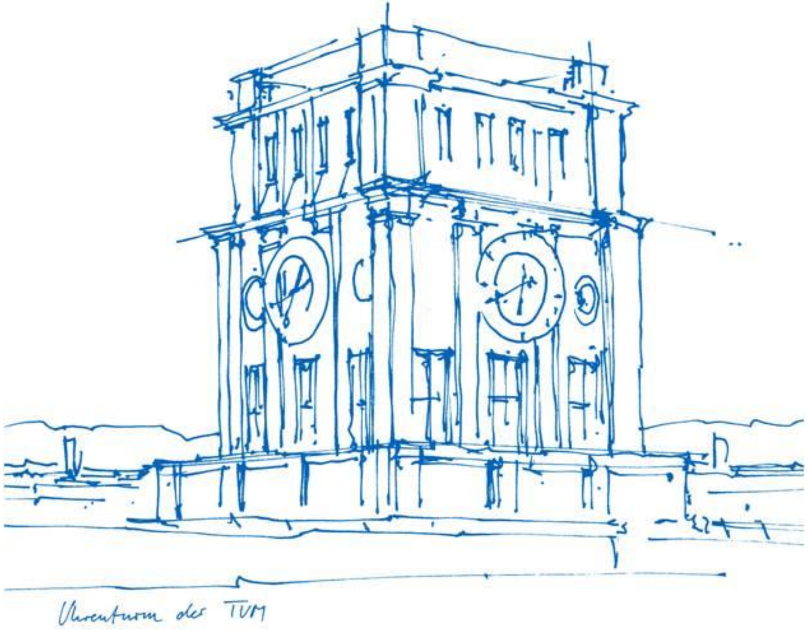


Table of contents

Algorithmic Pipeline for Graph Drawing Optimization

Baseline Approach

All strategies implement the **Simulated Annealing** meta-heuristic to find a global minimum for the graph's energy function. The energy function minimizes edge crossings while maintaining graph structure.

Legacy (Baseline)

Algorithm: **Simulated Annealing** with additional forces:

Node Repulsion: Pushes nodes apart to prevent overlap (Force-Directed Graph Drawing principle).

Spring Attraction: Pulls connected nodes together to minimize edge length.

Technology: Pure Python.

Description: The original implementation. It calculates edge crossings and energy using standard Python loops.

Pros: Simple, easy to debug, serves as a correctness baseline.

Cons: Very slow for large graphs (O(E^2)) complexity in interpreted Python).

Strategy Combination

Hybrid Optimization Pipeline

Phase 1: Initialization (Structural Setup)

- Goal: Create a topologically sound "warm start" on the grid
- Engine: FMME (Fast Multipole Multilevel Embedder) for global layout
- Key Step: Collision-Free Grid Mapping, prioritizing the placement of high-degree anchor vertices

Strategy Combination

Hybrid Optimization Pipeline

Phase 2: Core Optimization (Targeted k Reduction)

- Goal: Use iterative search to aggressively reduce the maximum crossing count (k).
- Engine: Simulated Annealing (SA), navigating the discrete space.
- Cost Function: Soft-Max Energy to disproportionately penalize edges with high crossings, creating a smooth gradient.
- Targeting: Move set evolves from global Shift/Swap (high T) to the Bottleneck Heuristic (low T) for specific k-edge intervention.

Strategy Combination

Hybrid Optimization Pipeline

Phase 3: Efficiency & Polish

- Goal: Ensure millions of iterations are feasible and finalize the coordinates.
- Speed: Uniform Grid Spatial Hashing reduces crossing checks via ΔE (change in energy) evaluation.
- Final Step: Deterministic Hill Climbing removes all remaining trivial inefficiencies after SA terminates.

Algorithm Accelerate

Numba & CUDA Integration

Core Idea: Utilize specialized Python tools for Just-In-Time (JIT) compilation and parallel processing

Primary Tool: Numba JIT Compiler

- Used the @numba.njit decorator to translate critical numerical loops (e.g., in the Soft-Max Energy calculation) into fast, optimized machine code from Python.
- Uses the same logic as Legacy but compiles the hot loops (energy calculation) into optimized machine code at runtime.
- Bypasses the Python Interpreter overhead for the critical $O(E^2)$ loops.
- Technology: Python + Numba (Just-In-Time Compiler).

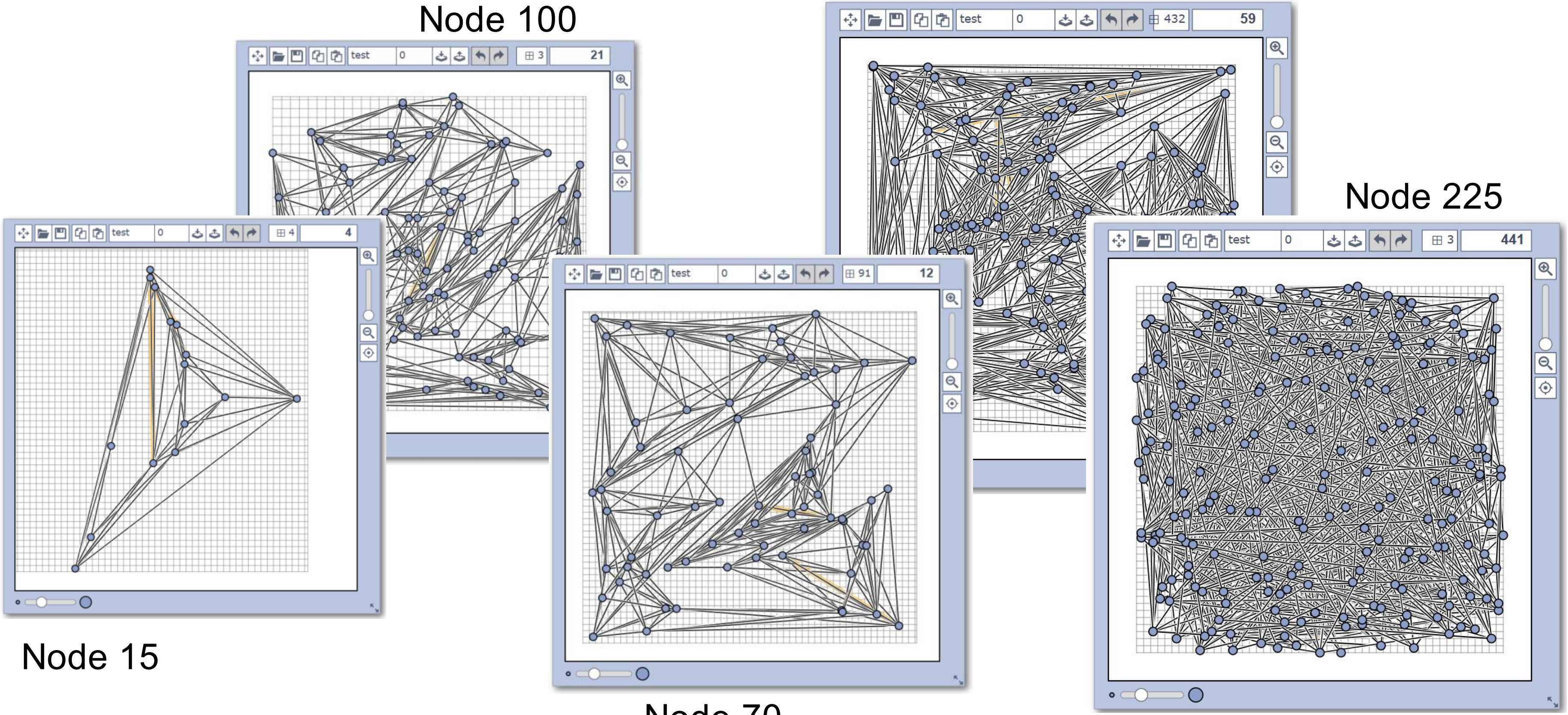
Algorithm Accelerate

Numba & CUDA Integration

Parallel Processing (CUDA)

- Distribute work across multiple CPU cores for general speedup.
- Utilized Numba with CUDA to offload the most intensive calculations (intersection checks via Delta Evaluation) directly to the GPU, achieving massive parallel speedup.
  - Massive Parallelism: Offloads the O(E^2) crossing counting task to the GPU.
  - Kernel Execution: Each GPU thread calculates the intersections for a single edge against all other edges in parallel.
  - Metropolis Update: The acceptance logic is handled efficiently after the heavy lifting is done on the GPU.
  - Technology: Python + CuPy + NVIDIA CUDA.
  - Description: Offloads the massively parallel task of counting edge crossings to the GPU.

Cases results



Node 100

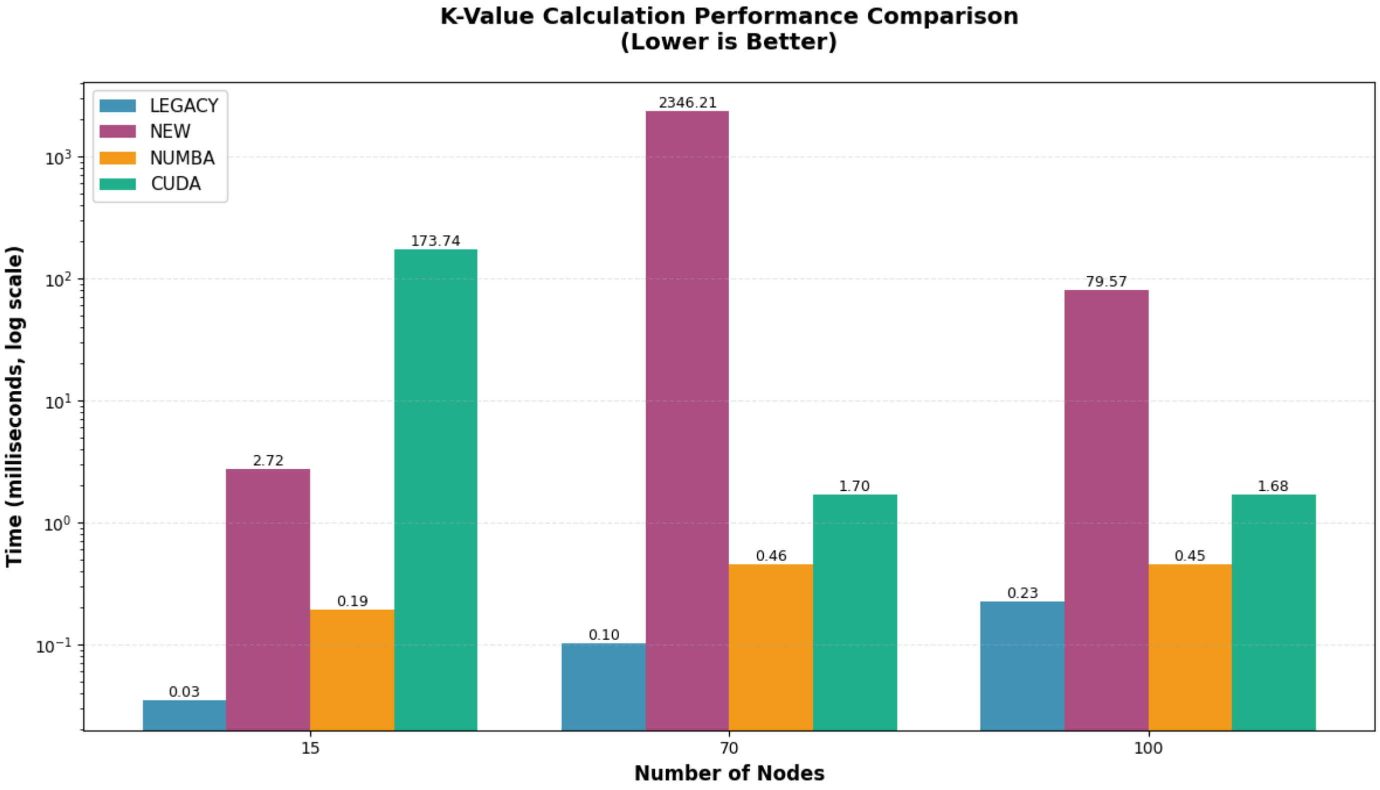
Node 150

Node 225

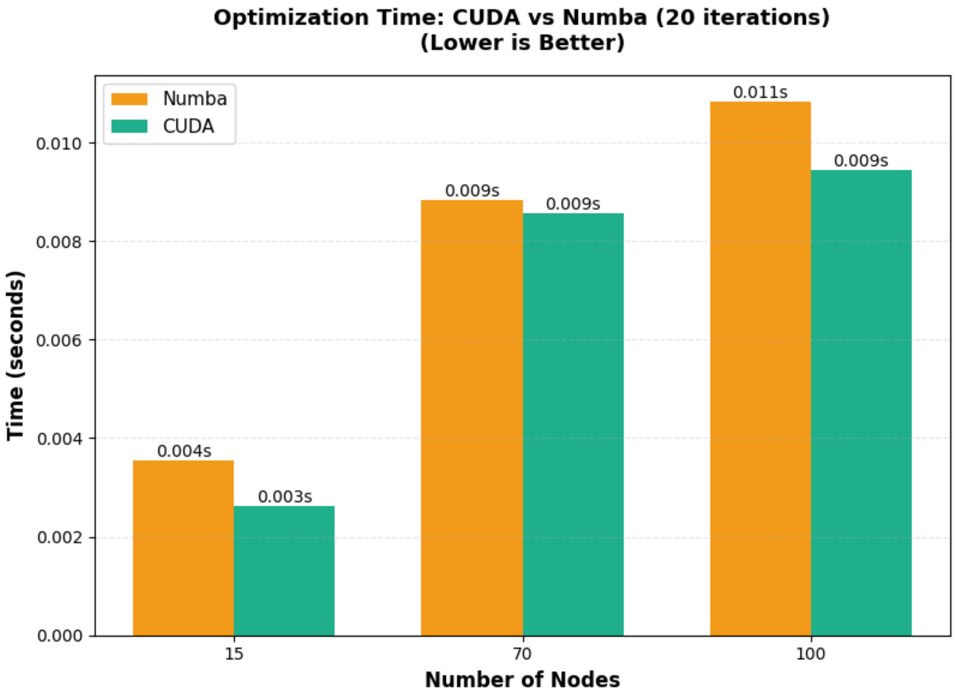
Node 15

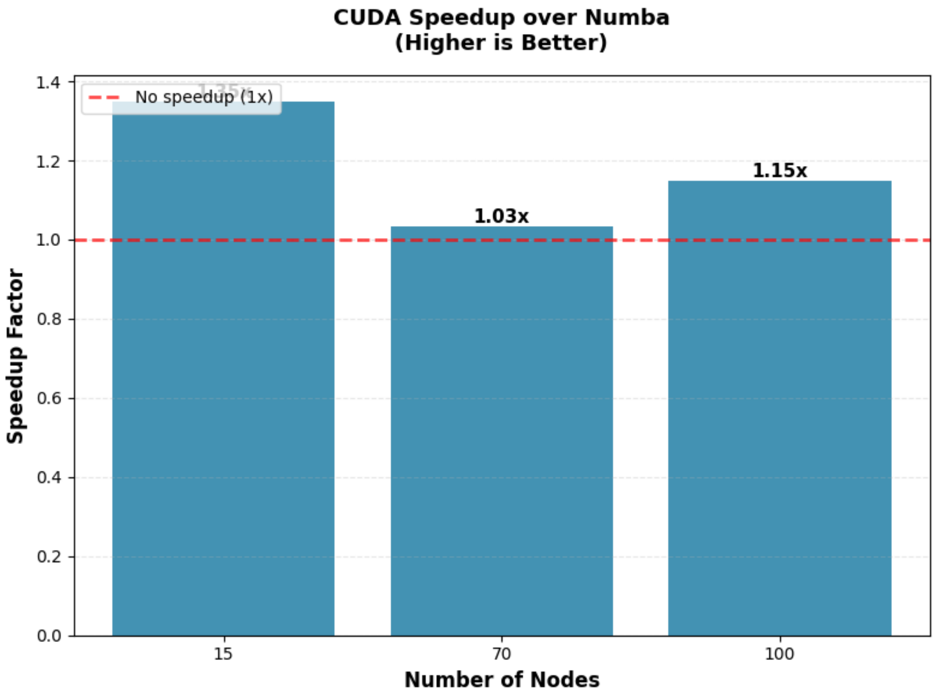
Node 70

Cases results

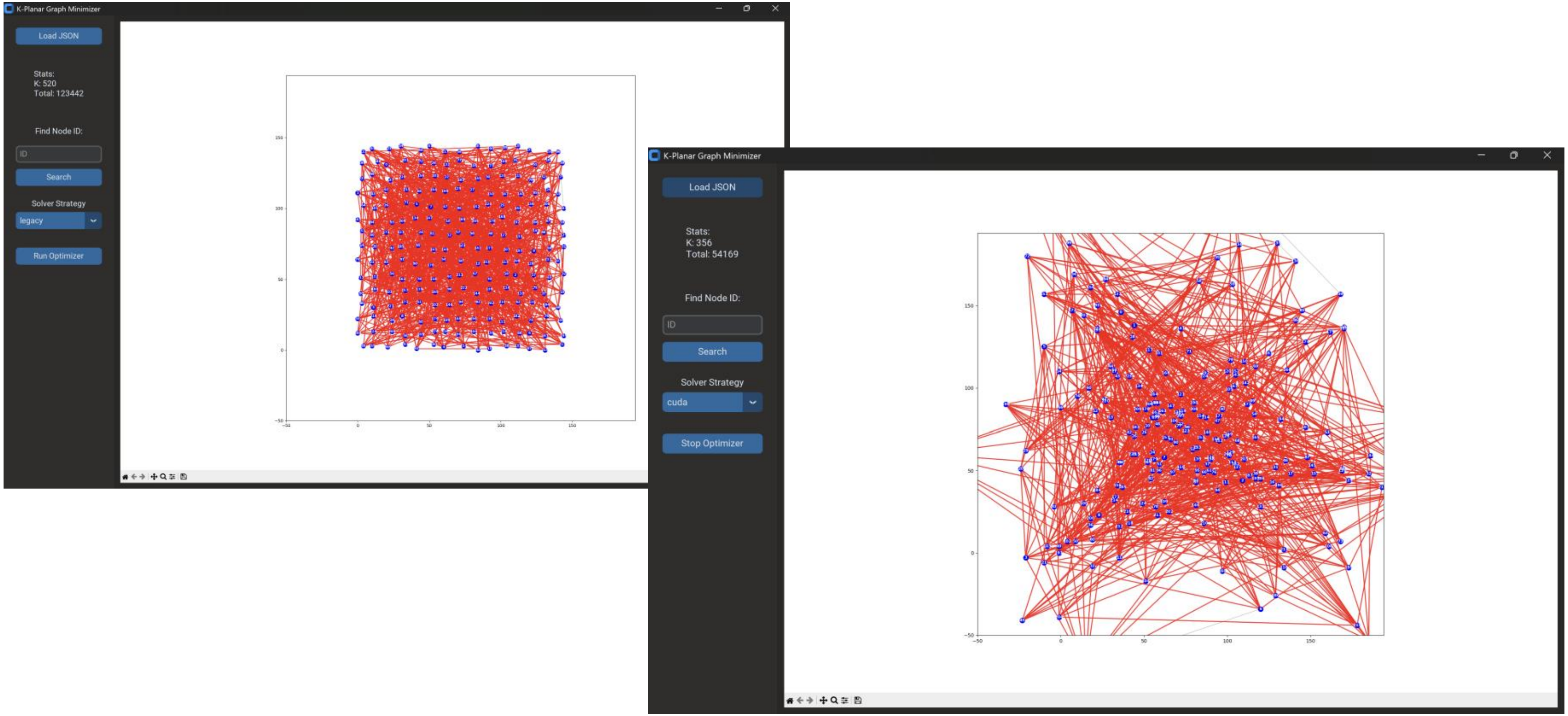


Cases results





Visualizer


K-Planar Graph Minimizer
Load JSON
Stats:
K: 520
Total: 123442
Find Node ID:
ID
Search
Solver Strategy
legacy
Run Optimizer
K-Planar Graph Minimizer
Load JSON
Stats:
K: 356
Total: 54169
Find Node ID:
ID
Search
Solver Strategy
cuda
Stop Optimizer

Visualizer

Layout

Sidebar (Left):

- **Controls**: Load JSON data, select solver strategy, search for nodes.
- **Stats Panel**: Real-time display of the K-value (max crossings per edge) and Total Crossings.

Main View (Right):

- **Interactive Plot**: Zoom, pan, and drag nodes manually.
- **Dynamic Scaling**: Automatically adjusts the view to fit the graph.

Visualizer

Edge Coloring (Visual Feedback)

The visualizer uses color to highlight problem areas in the graph:

Gray Edges: Low Conflict

These edges have **3 or fewer crossings**. They are considered "acceptable" in the current layout.

Condition: `crossings <= 3`

Red Edges: High Conflict.

These edges have **more than 3 crossings**.

Purpose: They visually indicate where the solver is struggling or where the graph is most tangled. The optimizer focuses on reducing these red edges to minimize the global energy.

Condition: `crossings > 3`

Outlook

Reinforcement Learning Approach

- Apply modifed RL Model created by TUM Heilbronn Chair of Efficient Algorithms:

  “Using Reinforcement Learning to Optimize the Global and Local Crossing Number”

- Speed up low Temperature Annealing by applying RL Node selection
  - Integration of low degree bias → Move Nodes with less potential for creating chaos

  → Improved Move Acceptance Rate (efficiency gain)

  → Focus on easy fixes, avoid high risk moves

References

1. Graph Drawing Contest Report - UPCommons,
https://upcommons.upc.edu/bitstream/handle/2117/425770/LIPIcs.GD.2024.41.pdf?sequence=1

2. [2510.00331] One-Sided Local Crossing Minimization - arXiv, https://arxiv.org/abs/2510.00331

3. One-Sided Local Crossing Minimization - arXiv, https://arxiv.org/html/2510.00331v1

4. One-Sided Local Crossing Minimization - ResearchGate, https://www.researchgate.net/publication/396094529_One-Sided_Local_Crossing_Minimization

5. TESTING k-PLANARITY IS NP-COMPLETE - MIT Mathematics, https://math.mit.edu/~urschel/publications/p2021a.pdf

6. Bottleneck Crossing Minimization in Layered Graphs - NC State University, https://techrep.csc.ncsu.edu/2010/TR-2010-13.pdf

7. Vertex Separation Problem - GRAFO Research Group, https://grafo.etsii.urjc.es/optsicom/mmacp.html

8. (PDF) Graph Drawing Contest Report - ResearchGate,
https://www.researchgate.net/publication/322620681_Graph_Drawing_Contest_Report

9. Graph Drawing Contest Report - DROPS, https://drops.dagstuhl.de/storage/00lipics/lipics-vol357-gd2025/LIPIcs.GD.2025.41/LIPIcs.GD.2025.41.pdf

10. OGDF – Open Graph Drawing Framework, https://ogdf.uos.de/

11. ogdf::FMMMLayout Class Reference - ogdf, https://ogdf.github.io/doc/ogdf/classogdf_1_1_f_m_m_m_layout.html

12. The Open Graph Drawing Framework (OGDF) - Brown Computer Science,
https://cs.brown.edu/people/rtamassi/gdhandbook/chapters/ogdf.pdf

References

13. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning |Operations Research - PubsOnLine, https://pubsonline.informs.org/doi/10.1287/opre.39.3.378

14. DRAWING GRAPHSUSING SIMULATED ANNEALING AND GRADIENT DESCENT, https://aile3.ijs.si/dunja/SiKDD2004/Papers/JanezBrank-GraphDrawing.pdf

15. Algorithms and Combinatorics for Beyond Planar Graphs - Universität Tübingen, https://ub01.uni-tuebingen.de/xmlui/bitstream/handle/10900/160137/MPfisterDissBeyondPlanarity-A.pdf?sequence=5&isAllowed=y

16.Bounding the pseudolinear crossing number of Kn via simulated annealing,https://kam.mff.cuni.cz/~balko/prezentace/Bounding_the_pseudolinear_crossing_number_of_kn_via_simulated_annealing.pdf

17. Stochastic Methods for One-Sided Bipartite Crossing Minimization and its Variants - UNM Digital Repository,https://digitalrepository.unm.edu/cgi/viewcontent.cgi?article=1064&context=cs_etds

18. Heuristics for the Min-Max ArcCrossing Problem in Graphs - Universitat de València, https://www.uv.es/~rmarti/paper/docs/gd12.pdf

19. CollisionDetection and Spatial Indexes - Andy Korth, https://kortham.net/posts/collision-detect-and-spatial-indexes/

20. When toUse Spatial Hashing vs Bounding Volume Hierarchy? - Game Development Stack Exchange,https://gamedev.stackexchange.com/questions/124186/when-to-use-spatial-hashing-vs-bounding-volume-hierarchy

21.1999 - HIP-Help Increase the Peace Program Manual 2nd Edition.pdf, https://afsc.org/sites/default/files/2023-06/1999%20-%20HIP-Help%20Increase%20the%20Peace%20Program%20Manual%202nd%20Edition.pdf