Winning the GD Challenge for the 4th Time: Our Approach

Julien Bianchetti
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France

Laurent Moalic
Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France

Abstract

We present the approach we designed to tackle and win the 2025 Graph Drawing Challenge on minimizing the k-planarity of graphs. Our method employs a multi-stage heuristic centered around two Simulated Annealing (SA) algorithms: the first aims to reduce the total number of crossings, while the second improves the k-value. To obtain a good initial solution, we first applied tools from the OGDF library, which helped reduce crossings. The challenge consisted of nine instances to optimize. Our approach achieved the best results on eight out of nine instances - sharing the top score twice with other teams and ranking first alone in six cases.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Human-centered computing → Graph drawings; Mathematics of computing → Combinatorial optimization

Keywords and phrases Graph Drawing Contest, simulated annealing, k-planarity

Digital Object Identifier 10.4230/LIPIcs.GD.2025.43

Category Graph Drawing Contest Abstract

1 Introduction

The 2025 Graph Drawing Contest live challenge addresses the problem of minimizing the parameter k in k-planar straight-line drawings of graphs. Given an input graph, the task is to compute a drawing in the plane where each edge is crossed by at most k other edges, and the objective is to minimize this maximum crossing number.

The algorithm we developed, named SAkGD, is based on two distinct Simulated Annealing (SA) phases, each with a specific fitness function. SA is a powerful metaheuristic for combinatorial optimization [7, 1] and has been successfully applied to aesthetic graph drawing in the past [3]. A key feature of our approach is that while each SA phase has its own parameters, they are not instance-dependent; we applied the same algorithm and parameters to every instance of the challenge.

2 General presentation of SAkGD: the SA based algorithm for k-planarity

The proposed algorithm is based on three main steps summarized as fallows:

1. Generation of a «not so bad» starting solution with OGDF
2. Improve the starting solution by removing as many crossing edges as possible
3. Optimize the improved solution in terms of k-value

2.1 Step 1: Generating a Not-So-Bad Starting Solution

For this step, our goal was to quickly generate a high-quality initial layout. We utilized the powerful Open Graph Drawing Framework (OGDF) [2], which provides robust implementations of several layout algorithms. We specifically used Stress Minimization [4] and the Fast Multipole Multilevel Method (FMMM) [6, 5].

Stress Minimization is applied only once, since it is deterministic – that is, two runs produce exactly the same result. Then, we apply FMMM as many times as possible within the first minute. Because FMMM is stochastic, two runs yield different solutions.

At the end of the first minute, or after the time required for Stress Minimization (SM) and FMMM, we consider the best solution obtained so far among the initial solution (directly from the data file), the one from SM, and all the FMMM runs. Here, the best solution is defined as the one with the smallest k-value.

2.2 Step 2: A 10-Minute SA for Crossing Reduction

The 2023 and 2024 contest topics focused on minimizing the total number of edge crossings in a graph. Another difference was that nodes could only be placed at predefined positions (locations). It is quite clear that reducing the number of crossing edges is relevant in terms of k-planarity. Indeed, even though decreasing the number of crossings is not necessarily synonymous with decreasing the k-value, it generally provides significant improvement in most cases.

Algorithm 1 describes the key steps for reducing the total number of edge crossings.

Algorithm 1 SA-based approach for removing edge crossings

Input: $initTemp$ the starting temperature, $decreaseTemp$ the decreasing temperature after each step, $decreaseTempWave$ the decreasing temperature after each wave, $tempLimit$ temperature limit before stopping, s_0 the initial solution
Output: $s*$ the best solution found so far

$s* \leftarrow s_0$
$k \leftarrow$ the starting k value from s_0
$crossing \leftarrow$ the total number of crossings from s_0
$kBest \leftarrow k$, $crossingBest \leftarrow crossing$
$startingTemp \leftarrow initTemp$
while $k > 0$ and $startingTemp > tempLimit$ **do**
  $currentTemp \leftarrow startingTemp$
  while $k > 0$ and $currentTemp > tempLimit$ **do**
    $node \leftarrow$ selectNode()
    $newPlace \leftarrow$ selectPlace()
    if acceptMove($node$, $newPlace$)==true **then**
      move($node$, $newPlace$) {update $crossing$ and k}
      if $k < kbest$ or $k == kbest$ and $crossings < crossingsBest$ **then**
        $kBest \leftarrow k$, $crossingBest \leftarrow crossing$, $s* \leftarrow currentSol$
      end if
    end if
  end while
  $startingTemp \leftarrow startingTemp * decreaseTemp$
  $currentSol \leftarrow s*$
end while
return $s*$

selectNode() and *selectPlace*() play a significant role. *selectNode*() is important because not all nodes have the same impact on the current k-value. If all edges originating from a given node have few crossings, moving that node will yield only a small improvement compared to moving a node connected to edges with many crossings. For this reason, nodes belonging to edges with many crossings are given a higher probability of being selected for movement.

selectPlace() is also important. If the new position selected for a node is far from its previous location, there is a risk of creating many new crossings. For this reason, we decided to give a higher probability of selection to positions close to the node's previous location.

A move is always accepted from *acceptMove*() if it reduces or maintains the total number of crossings. A move that increases the crossings by ΔC is accepted with a probability of $\exp(-\Delta C/T_{current})$.

In this phase, we focus on optimizing the total number of crossings. However, it should be noted that each new wave starts from the best k-solution found so far. This is done to prepare for the final and main step of the proposed algorithm.

2.3 Step 3: Remaining Time for k-Value Optimization

With the remaining time (at most 49 minutes), we switch to a second SA algorithm that directly optimizes the k-value. This phase uses the same structure as Algorithm 1 but employs a more sophisticated fitness function for the *acceptMove*() criterion:

1. Primary Objective: Local k-value Improvement. A move is evaluated based on the change in the maximum number of crossings on any edge involved in the move (i.e., edges incident to the moved node and edges that cross them). This local computation is much faster than re-evaluating the global k-value.
2. Secondary Objective: Total Crossing Number. If a move does not change the local maximum k-value, the decision is based on the change in the total number of crossings, as in the previous phase.

A move is accepted if it improves the solution according to this objective. Worsening moves are still accepted based on the same probabilistic formula, using the fitness change as ΔE. This dual-objective approach allows the search to fine-tune the placement of nodes that are bottlenecks for the k-value, while still encouraging a general reduction in crossings.

3 Application, parameters and results on the 2025 GA contest dataset

Table 1 presents the parameters used during the contest. These were not tuned per-instance, demonstrating the robustness of the approach. It could be interesting to tune them, especially based on instances characteristics.

Table 1 SA parameters for minimizing the total crossings number and the k-value

	initTemp	*decreaseTemp*	*decreaseTempWave*	*tempLimit*
min cross	50.0	0.999	0.99	0.01
min k	1.0	0.9999	0.99	0.01

Our method performed exceptionally well, securing the top score on 8 of the 9 contest instances. Table 2 summarizes these results.

Table 2 Performance on the 2025 Graph Drawing Contest dataset.

Instance	Our k-value	Best k-value	2nd Best k-value	3rd Best k-value	Our Rank
graph 1	**9**	9	9	9	1st (Shared)
graph 2	**3**	3	3	3	1st (Shared)
graph 3	**34**	34	40	43	1st
graph 4	**6**	6	7	8	1st
graph 5	**76**	76	81	85	1st
graph 6	**568**	560	568	1081	2nd
graph 7	**31**	31	32	40	1st
graph 8	**15**	15	20	29	1st
graph 9	**12**	12	16	21	1st

4 Conclusion

We presented SAkGD, the multi-stage heuristic that won the 2025 Graph Drawing Challenge. Its success stems from a three-step process: a fast generation of a good initial layout using OGDF, followed by two specialized Simulated Annealing phases. The first phase effectively reduces the total crossing number, creating a promising search space, while the second phase aggressively targets the primary objective of minimizing the k-value using a fitness function. The use of instance-independent parameters highlights the general applicability of our method. Future work could explore adaptive parameter tuning based on graph properties or hybridizing the SA with other local search techniques.

References

1 Dimitris Bertsimas and John Tsitsiklis. Simulated Annealing. *Statistical Science*, 8(1):10–15, 1993. doi:10.1214/ss/1177011077.

2 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (ogdf). *Chapter 17 in: R. Tamassia (ed.), Handbook of Graph Drawing and Visualization, CRC Press.*, 2014.

3 Ron Davidson and David Harel. Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.*, 15(4):301–331, October 1996. doi:10.1145/234535.234538.

4 Emden Gansner, Yehuda Koren, and Stephen North. Graph drawing by stress majorization. In *Graph Drawing*, volume 3383, pages 239–250, September 2004. doi:10.1007/978-3-540-31843-9_25.

5 Martin Gronemann. Engineering the Fast-Multipole-Multilevel method for Multicore and SIMD architectures. Master's thesis, Technische Universität Dortmund, Dortmund, Germany, 2009.

6 Stefan Hachul and Michael Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *Graph Drawing*, volume 3383, pages 294–305, September 2004. doi:10.1007/978-3-540-31843-9_29.

7 S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. doi:10.1126/science.220.4598.671.