UzL Heuristic Solver for One-Sided Crossing Minimization

Max Bannach
European Space Agency

Florian Chudigiewitsch
Institute for Theoretical Computer Science, University of Lübeck, Germany

Kim-Manuel Klein
Institute for Theoretical Computer Science, University of Lübeck, Germany

Till Tantau
Institute for Theoretical Computer Science, University of Lübeck, Germany

Marcel Wienöbst[1]
Institute for Theoretical Computer Science, University of Lübeck, Germany

Abstract

This document contains a short description of our solver *sisyphus* for the one-sided crossing minimization problem that we submitted to the heuristic track of the PACE challenge 2024. After performing the well-known reduction to feedback arc set, each strongly connected component is initially ordered by inserting the vertices one by one at their best position in the current ordering of so-far considered vertices. The obtained initial ordering is further improved by a hill climber based on the sifting strategy. This approach is run repeatedly and the overall best solution is output.

2012 ACM Subject Classification Theory of computation → Randomized local search

Keywords and phrases hill climbing, local search, feedback arc set, crossing minimization

Supplementary Material Code repository
URL: `https://github.com/mwien/sisyphus`
DOI: `10.5281/zenodo.11533177`

1 Introduction

One-sided crossing minimization is a fundamental problem in graph drawing. For a given bipartite graph $G = (V_1 \cup V_2, E)$ and a linear ordering τ of V_1, the goal is to find a linear ordering π of V_2 that minimizes the number of *crossings*, that is $(\{u_1, u_2\}, \{v_1, v_2\}) \in E^2$ such that $\tau^{-1}(u_1) < \tau^{-1}(v_1)$ and $\pi^{-1}(u_2) > \pi^{-1}(v_2)$, with $\tau^{-1}(x)$ and $\pi^{-1}(x)$ giving the position of vertex x in the respective ordering. Vice versa $\pi(i)$ and $\tau(i)$ denote the vertex at position i in π and τ, respectively.

The cost of putting a vertex at a certain position can be expressed by sums of *crossing numbers*: c_{uv} gives the number of crossings of edges incident to u or v if u would be ordered to the left of v. Hence, the crossings incurred by having vertex u at position i in linear ordering π is given by

$$\sum_{j=1}^{i-1} c_{\pi(j)u} + \sum_{j=i+1}^{n} c_{u\pi(j)}.$$

This enables *insertion-based* heuristics, where the vertices are added one by one, each at the position that introduces the least number of crossings, and *sifting* heuristics [1] that

[1] Corresponding author.

reinsert vertices at another position in the ordering if it reduces the overall number of crossings. In both cases, the best position can be computed in linear time in the length of the ordering, given that the crossing numbers were computed beforehand. In the following, we combine these heuristic strategies in our approach.

2 Description of Our Approach

Our solver first uses the well-known reduction of the problem to feedback arc set, which allows for solving each strongly connected component separately. For each such component, we solve the linear ordering problem stated above.

The solver performs repeated hill-climber runs, which consist of the following steps:

1. Find an initial linear ordering of the component using an insertion-based heuristic.
2. Improve this solution through sifting steps, i.e., reinserting a vertex at a "better" position.

The initial insertion-based heuristic iterates over the vertices in a random order and, as stated above, inserts each vertex at its current best position (or, more precisely, one of the best positions, in case there are multiple).

Afterwards, a vertex is repeatedly chosen at random and reinserted at a position that reduces the number of crossings by the maximum amount. In case the number of crossings would increase, the vertex remains at its current position. This procedure terminates if there is no improvement during several steps. The overall best ordering is output at the time limit.

There are a few additional details that improve the performance further. First, we perform a sifting hill-climber run after every 50 insertions during the insertion-based heuristic. The rationale is that the insertion is better if the current ordering is closer to the optimum. Second, after the first 60 seconds, we check, in the equivalent feedback arc set formulation of the problem, whether there are edges that would always have been part of the solution. These are (temporarily) removed, and the strongly connected components of the resulting subgraph are computed. In the remaining time, the heuristic solver is run on these components, which are often much smaller.

Finally, for large instances (more than 10^4 vertices in $|V_2|$), we adapt our solver so that it does not perform the reduction step to the strongly connected components of the feedback arc set instance. Here, starting with the well-known barycenter heuristic [2], the sifting hill-climber is started immediately and it performs only a single run, i.e., it is never restarted.

References

1 Christian Matuszewski, Robby Schönfeld, and Paul Molitor. Using sifting for k-layer straightline crossing minimization. In *Graph Drawing: 7th International Symposium, GD99 Štiřín Castle, Czech Republic September 15–19, 1999 Proceedings 7*, pages 217–224. Springer, 1999.

2 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.