weberknecht

Johannes Rauch

Ulm University, Institute of Optimization and Operations Research, Germany

Abstract

This article describes the implementation of weberknecht(_h)[1], a solver for One-Sided Crossing Minimization that participated in the Parameterized Algorithms and Computational Experiments Challenge 2024.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases One-Sided Crossing Minimization

Supplementary Material !!!!!!!!!!!!!!

Acknowledgements I thank Henning Bruhn-Fujimoto and Dieter Rautenbach for helpful discussions.

1 Preliminaries

An instance $(G = (A, B, E), \pi_A)$ of One-Sided Crossing Minimization is a bipartite graph G with n vertices, bipartition sets A and B, and a linear ordering π_A of A. The goal is to find a linear ordering π_B of B that minimizes the number of crossing edges if the graph were to be drawn in the plane such that

- the vertices of A and B are on two distinct parallel lines, respectively, and
- the order of the vertices of A and B on the lines is consistent with π_A and π_B, respectively.

We assume that $A = [n_0] := \{1, \ldots, n_0\}$ and $B = \{n_0 + 1, \ldots, n_0 + n_1\}$ for some positive integers n_0 and n_1. We think of π_A and π_B as bijections $A \to [n_0]$ and $B \to [n_1]$, respectively. If $\pi_B(u) < \pi_B(v)$ for $u, v \in B$, we say that u is ordered before v, or u is to the left of v.

Let $c_{u,v}$ denote the number of crossings of edges incident to $u, v \in B$ if $\pi_B(u) < \pi_B(v)$. A mixed-integer program for One-Sided Crossing Minimization is given by

$$\begin{aligned} \text{minimize} \quad & \sum_{\substack{u,v \in B \\ u<v}} (c_{u,v} - c_{v,u}) \cdot x_{u,v} + \sum_{\substack{u,v \in B \\ u<v}} c_{v,u} \\ \text{subject to } 0 \le x_{u,v} + x_{v,w} - x_{u,w} \le 1 \quad & \text{for all } u, v, w \in B, u < v < w, \\ x_{u,v} \in \{0, 1\} \quad & \text{for all } u, v \in B, u < v. \end{aligned} \tag{P_I}$$

So, u is ordered before v if and only if $x_{u,v} = 1$ for $u, v \in B, u < v$.

2 Overview

The solver weberknecht(_h) is written in C++. First, the exact solver weberknecht runs the uninformed and improvement heuristics described in Section 3. Then it applies the data reduction rules described in Section 4. Last, it solves a reduced version of the mixed-integer program associated to the input instance with a custom branch and bound and cut algorithm described in Section 5. The heuristic solver weberknecht_h only runs the uninformed and improvement heuristic (except the local search heuristic).

[1] Weberknecht is the german name for the harvestman spider. It is a composite word consisting of the words Weber = weaver and Knecht = workman.

3 Heuristics

We distinguish between uninformed and informed heuristics, which build a solution from the ground up, and improvement heuristics, which try to improve a given solution. Due to the reduction rules we may assume from here that there are no isolated vertices in G.

Uninformed Heuristics. The uninformed heuristics order the vertices of B such that the *scores* $s(v)$ of vertices $v \in B$ is non-decreasing:

- In the *barycenter heuristic*, we have $s(v) = \frac{1}{d_G(v)} = \sum_{u \in N_G(v)} u$ (recall that $A = [n_0]$). Eades and Wormald [4] proved that this method has an $\mathcal{O}(\sqrt{n})$ approximation factor, which is best possible up to a constant factor under certain assumptions.
- Let $d = d_G(v)$ and let $\{w_0, \ldots, w_{d-1}\}$ be the neighbors of v in G with $w_0 < \cdots < w_{d-1}$. In the *median heuristic*, the score of v is $s(v) = w_{(d-1)/2}$ if d is odd and $s(v) = (w_{d/2-1} + w_{d/2})/2$ if d is even. Eades and Wormald [4] proved that this method is a factor three approximation algorithm.
- In the *probabilistic median heuristic*, we draw a value x from $[0.0957, 0.9043]$ uniformly at random, and the score of v is then $s(v) = w_{\lfloor x \cdot d \rfloor}$. This is essentially the approximation algorithm of Nagamochi [7], which has an approximation factor of 1.4664 in expectancy.

Informed Heuristics. The informed heuristics get a fractional solution of the linear program relaxation of (P_I) as an additional input.

- The *sort heuristic* works like a uninformed heuristics. The score for vertex $v \in B$ is $s(v) = \sum_{u \in B, u<v} x_{u,v} + \sum_{u \in B, v<u} (1 - x_{v,u})$.
- Classical *randomized rounding heuristic*.
- *Relaxation induced neighborhood search* [1].

Improvement Heuristics. Assume that $\pi_B = u_1 u_2 \ldots u_{n_1}$ is the current best solution.

- The *shift heuristic* that Grötschel et al. [5] describes tries if shifting a single vertex improves the current solution.
- In the *local search heuristic*, we solve a reduced version of (P_I) to optimality, where we only add variables x_{u_i,u_j} with $|i - j| < w$ for some parameter w.

4 Data Reduction

The solver weberknecht implements the following data reduction rules:

- Vertices of degree zero in B are put on the leftmost positions in the linear ordering π_B.
- Let l_v (r_v) be the neighbor of $v \in B$ in G that minimizes (maximizes) π_A, respectively. Dujmović and Whitesides [3] noted that, if there exists two nonempty sets $B_1, B_2 \subseteq B$ and a vertex $q \in A$ such that for all $v \in B_1$ we have that $\pi_A(r_v) \leq \pi_A(q)$, and for all $v \in B_2$ we have that $\pi_A(q) \leq \pi_A(l_v)$, then the vertices of B_1 appear before the vertices B_2 in an optimal solution. In this case we can split the instance into two subinstances.
- Dujmović and Whitesides [3] proved that, if π_B is an optimal solution, and $c_{u,v} = 0$ and $c_{v,u} > 0$, then $\pi_B(u) < \pi_B(v)$.
- Dujmović et al. [2] described a particular case of the next reduction rule. Let $c_{u,v} < c_{v,u}$. We describe the idea with the example in Figure 1. Imagine that we draw some edge $x_i y_j$ into Figure 1. If the number of edges crossed by $x_i y_j$ on the left side is at most the number of edges crossed by $x_i y_j$ on the right side for all edges of the form $x_i y_j$, then we have $\pi_B(u) < \pi_B(v)$ in any optimal solution π_B: Otherwise we could improve the solution by simply exchanging the positions of u and v. Note that this reduction rule is only applicable if $d_G(u) = d_G(v)$ as witnessed by $x_2 y_1$ and $x_2 y_k$ ($k = 5$ here).

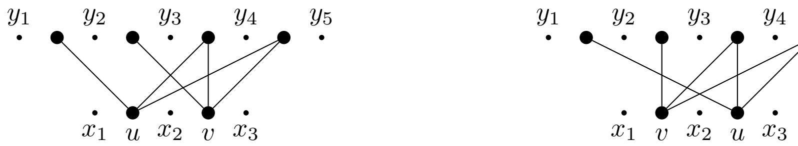

Figure 1

- The value $\ell b = \sum_{u,v\in B, u<v} \min(c_{u,v}, c_{v,u})$ is a lower bound on the number of crossings of an optimal solution. Suppose that we have already computed a solution with ub crossings. Then, if $c_{u,v} \geq ub - \ell b$ for some $u, v \in B$, it suffices to only consider orderings π_B with $\pi_B(u) > \pi_B(v)$ for the remaining execution.
- After the execution of the described reduction rules, some variables $x_{u,v}$ of (P_I) have a fixed value due to the constraints.

5 Branch and Bound and Cut

The solver weberknecht implements a rudimentary branch and bound and cut algorithm. We use HiGHS [6] only as a linear program solver since it does not (yet) implement lazy constraints. To avoid adding all $\Theta(n^3)$ constraints, we solve the linear program relaxation of (P_I) as follows.

1. Create a linear program (P) with the objective function of (P_I) and no constraints.
2. Solve (P).
3. If the current solution violates constraints of (P_I), add them to (P) and go to 2.

Let ub denote the number of crossings of the current best solution. Then, until we have a optimal solution, weberknecht does the following:

1. Solve (P) with the method described above.
2. If (P) is infeasible, backtrack.
3. If the rounded objective value of P is at least ub, backtrack.
4. If the current solution of (P) is integral, update the best solution and backtrack.
5. Run informed heuristics and branch.

References

1. Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102:71–90, 2005.
2. Vida Dujmović, Henning Fernau, and Michael Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. *Journal of Discrete Algorithms*, 6(2):313–323, 2008.
3. Vida Dujmović and Sue Whitesides. An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. *Algorithmica*, 40:15–31, 2004.
4. Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11:379–403, 1994.
5. Martin Grötschel, Michael Jünger, and Gerhard Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations research*, 32(6):1195–1220, 1984.
6. Qi Huangfu and J. A. Julian Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.
7. Hiroshi Nagamochi. An improved bound on the one-sided minimum crossing number in two-layered drawings. *Discrete & Computational Geometry*, 33:569–591, 2005.