PACE Solver Description: OCMu64, a Solver for One-Sided Crossing Minimization

Ragnar Groot Koerkamp
ETH Zurich, Switzerland

Mees de Vries
Unaffiliated, The Netherlands

Abstract

Given a bipartite graph (A, B), the *one-sided crossing minimization* (OCM) problem is to find an ordering of the vertices of B that minimizes the number of edge crossings when drawn in the plane.

We introduce the novel *strongly fixed*, *practically fixed*, and *practically glued* reductions that maximally generalize some existing reductions. We apply these in our exact solver `OCMu64`, that directly uses branch-and-bound on the ordering of the vertices of B and does not depend on ILP or SAT solvers.

2012 ACM Subject Classification Theory of computation → Mathematical optimization; Theory of computation → Computational geometry

Keywords and phrases Graph drawing, crossing number, branch and bound

Digital Object Identifier 10.4230/LIPIcs.IPEC.2024.35

Supplementary Material *Software (Source Code)*: `https://github.com/mjdv/ocmu64` [6]
Software (Source Code): `https://doi.org/10.5281/zenodo.11671980` [7]

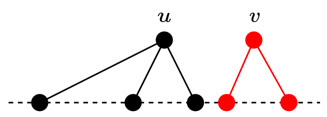
1 Introduction

The 2024 edition of PACE, an annual optimization challenge, considers the *one-sided crossing minimization* problem, defined as follows. Given is a bipartite graph (A, B) that is drawn in the plane at points $(i, 0)$ and $(j, 1)$ for components A and B respectively. The ordering of A is fixed, and the goal is the find an ordering of B that minimizes the number of crossings when edges are drawn as straight lines. We introduce some new reductions and give an overview of our algorithm. Proofs are brief or omitted due to lack of space.

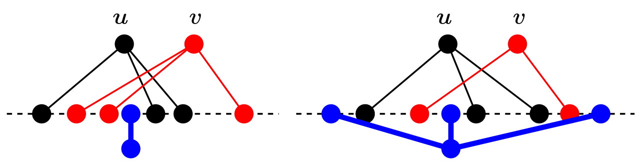
2 Definitions

We use $<$ to compare vertices in A in their fixed ordering. We generalize to weighted graphs for the proof of Lemma 3: a node $u \in B$ is taken to be a function $u : A \to \mathbb{R}^{\geq 0}$, where weight 0 represents an absent edge. Write $N(u) = \operatorname{supp}(u) \subseteq A$ for the set of neighbors of u. We write $W_u = \sum_{a \in A} u(a)$ for the total weight of u, and set $\bar{u} = u/W_u$. We think of $\bar{u}$ as a probability distribution, and also consider the cumulative distribution function $F_{\bar{u}}(b) = \sum_{a \leq b} \bar{u}(a)$. We write $c(u, v) = \sum_{(a,b) \in A^2} u(a)v(b)[a > b]$; in the unweighted case, this is the *crossing number*, the number of crossings between edges incident to u and v when u is drawn before v. For $X, Y \subseteq B$ we set $c(X, Y) = \sum_{x \in X} \sum_{y \in Y} c(x, y)$ for the cost of ordering all vertices of X before all vertices of Y. More generally, $c(X, Y, Z) = c(X, Y) + c(X, Z) + c(Y, Z)$. We also consider the *reduced cost* $r(X, Y) = c(X, Y) - c(Y, X)$, which is negative when X is better *before* Y and positive when X is better *after* Y.

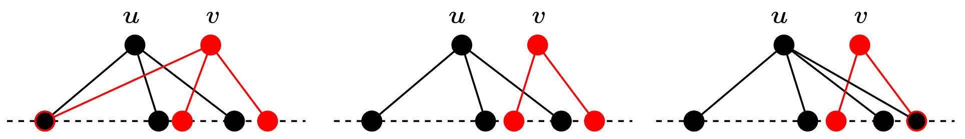
We write $u \prec v$ when u must come before v in all minimal solutions, and say that (u, v) is a *fixed* pair.



(a) An instance of Observation 1.



(b) Examples of blocking sets (Lemma 7), preventing $u \prec v$.



(c) Examples of strongly fixed pairs (Definition 2).

Figure 1 Various examples of $u, v \in B$, whose neighbors neighbours in A (on the dashed line) have fixed positions.

3 Methods

3.1 Reductions

Fixed pairs. To reduce the search space of all possible orderings of B, it is crucial to automatically find as many fixed pairs in B as possible. Ideally, one would be able to determine whether $u \prec v$ by inspecting only u, v. For example, the following result, shown in Figure 1a is well known [2, Lemma 1], [1, RR1].

▶ **Observation 1.** *Let $u, v \in B$. When $\max N(u) \leq \min N(v)$ and $\bar{u} \neq \bar{v}$, then $u \prec v$.*

We give a much stronger version of Observation 1, with some examples in Figure 1c. In fact, Lemma 3 is the strongest generalization possible when only considering u and v themselves, without further inspection of A or the other elements of B (see Remark 6). This also generalizes RR3 of [1].

▶ **Definition 2** (Strongly fixed pair)**.** *We call $u, v \in B$ a* strongly fixed pair *if for every $b \in A$ we have $F_{\bar{u}}(b) \geq F_{\bar{v}}(b)$, and at least one of these inequalities is strict. Note that this implies $r(u, v) < 0$.*

▶ **Lemma 3.** *When u, v is a strongly fixed pair, then for any $w : A \to \mathbb{R}^{\geq 0}$ we have*

$$r(w, \bar{v}) \leq r(w, \bar{u}).$$

Proof. Consider the least element $a_0 \in A$ such that $\bar{u}(a_0) \neq \bar{v}(a_0)$. We must have $\bar{u}(a_0) > \bar{v}(a_0)$. Now consider the transformation of $\bar{u}$ which "moves" $\delta := \bar{u}(a_0) - \bar{v}(a_0)$ of weight from a_0 to its successor $a_1 \in A$, and call this transformed function $\bar{u}'$. Then

$$r(w, \bar{u}') = r(w, \bar{u}) - \delta w(a_0) - \delta w(a_1) \leq r(w, \bar{u}).$$

Since $\bar{v}$ is obtained from $\bar{u}$ by a sequence of such transformations, the inequality follows. ◀

▶ **Lemma 4.** *If (u, v) is strongly fixed, then $u \prec v$.*

Proof. Suppose towards a contradiction that $v < x_0 < \cdots < x_k < u$ is part of an optimal solution. Write $X = \sum_i x_i$ for the combined function. Then by assumption $r(X, u) \leq 0$, and therefore $r(X, v) = W_v r(X, \bar{v}) \leq W_v r(X, \bar{u}) = W_v / W_u \cdot r(X, u) \leq 0$. But then $c(X, u, v) < c(X, v, u) \leq c(v, X, u)$, which contradicts (v, X, u) being optimal. ◀

▶ Remark 5. Consider $u \neq v$ from the original, unweighted problem, taking only values $0, 1$. Let $n = |N(u)|, m = |N(v)|$, and consider both as ordered lists. Then u, v are strongly fixed if and only if for all $0 \leq i < n$, $N(u)_i \leq N(v)_{\lfloor i \cdot m/n \rfloor}$ and at least one of the inequalities is strict, which is how we check in practice whether u, v is strongly fixed.

▶ Remark 6. Suppose that u, v with $\bar{u} \neq \bar{v}$ are not strongly fixed, and let $b \in A$ be such that $F_{\bar{u}}(b) < F_{\bar{v}}(b)$. We would like to construct a node $X \in B$ such that $c(v, X, u) < \min(c(X, u, v), c(u, v, X))$, so that $v \prec u$. Suppose that there are nodes $l, m, r \in A$, with m sitting between b and its successor in A, and $l < (\operatorname{supp}(u) \cup \operatorname{supp}(v)) < r$ (see Figure 1b). Then let X be connected to l, m, r. By choosing the weights of X appropriately, we can let both $r(X, v) = W_v r(X, \bar{v})$ and $r(u, X) = W_u r(\bar{u}, X)$ grow arbitrarily large. Since we can scale up the weights of X, it suffices to make $r(X, \bar{v})$ and $r(\bar{u}, X)$ simultaneously positive.

Now $r(X, \bar{v}) = (2F_{\bar{v}}(b) - 1)X(m) + (X(r) - X(l))$, and $r(\bar{u}, X) = (1 - 2F_{\bar{u}}(b))X(m) - (X(r) - X(l))$. By choosing the weights $X(r), X(l)$ appropriately, we can make these two values equal, so it suffices if their *average* is positive, and indeed

$$(r(X, \bar{v}) + r(\bar{u}, X))/2 = X(m)(F_{\bar{v}}(b) - F_{\bar{u}}(b)) > 0.$$

This implies $c(v, X, u) < \min(c(X, u, v), c(u, v, X))$, and thus demonstrates that if one wants to show that $u \prec v$, when u, v are not strongly fixed, one must consider features of the graph other than u, v and their neighbours. In other words, Lemma 3 is optimal. Note that this remark also applies in the unweighted case, by taking l, m, r to be sets of nodes rather than single nodes with weighted edges.

Although such a node X may exist in theory, it does not have to exist in the actual set B, motivating the following definition that generalizes RRLO2 of [1].

▶ **Lemma 7.** *Suppose $r(u, v) < 0$. A* blocking set *$X \subseteq B - \{u, v\}$ is a set such that $c(v, X, u) \leq \min(c(v, u, X), (X, v, u))$. If there is no blocking set for (u, v), we call it a* practically fixed pair*, and $u \prec v$.*

In practice, such a blocking set X can be found, if one exists, using a knapsack-like algorithm: for each $x \in B - \{u, v\}$, add a point $P_x = (r(v, x), r(x, u))$, and search for a subset summing to $\leq (r(u, v), r(u, v))$. Figure 1b shows some examples.

Note that we do not require (v, X, u) to be a true local minimum, since we do not consider interactions between vertices in X, as that would make ruling out the existence of such sets much harder.

▶ Remark 8 (Weak variants). It is also possible to consider *weak* variants of the above lemmas that only imply that $u < v$ in *some* optimal solution. This requires careful handling of cycles like $u \preceq v \preceq w \preceq u$.

Gluing. We now turn our attention to *gluing*, i.e., proving that two vertices u and v always go right next to each other, and we can treat them as a single vertex. First, let us see that we cannot get a "strong gluing" result analogous to Lemma 4.

▶ Remark 9. When $N(u) = N(v)$ in the unweighted case, or more generally $\bar{u} = \bar{v}$, we can glue u and v. Otherwise when $r(u, v) \leq 0$, there is an $X : A \to \mathbb{R}^{\geq 0}$ such that (u, X, v) is strictly better than (u, v, X) and (X, u, v).

▶ **Lemma 10** (Practical gluing)**.** *Let u and v satisfy $r(u, v) \leq 0$. A non-empty subset $X \subseteq B - \{u, v\}$ is* blocking *when $c(u, X, v) \leq \min(c(u, v, X), c(X, u, v))$. If there is no blocking set, then we can glue u, v.*

Again such sets X can be found or proven to not exist using a knapsack algorithm: add points $P_x = (r(u,x), r(x,v))$ and search for a non-empty set summing to $\leq (0,0)$.

Let us also mention this gluing-like reduction: *gluing to the front*, implied by [1, RRL01].

▶ **Lemma 11** (Greedy). *When $r(u,x) \leq 0$ for all $x \in B$, there is a solution that starts with u.*

▶ Remark 12 (Tail variants). Our branch-and-bound method fixes vertices of the solution from left to right. Thus, at each step Lemmas 7 and 10 can be applied to just the *tail*.

3.2 Branch-and-bound

Our solver `OCMu64` is based on a standard branch-and-bound on the ordering of the solution. We start with fixed prefix $P = ()$ and tail $T = B$, and in each step we try (a subset of) all vertices in T as the next vertex appended to P. In a preprocessing step we compute the trivial lower bound $S_0 = \sum_{u,v} \min(c(u,v), c(v,u))$ [5, Lemma 4][2, Fact 3] on the score. We keep track of the score S_P of the prefix and $S_{PT} = c(P,T)$ of prefix-tail intersections, and abort when this score goes above the best solution found so far. The *excess* of a tail is its optimal score minus the trivial lower bound. We do a number of optimizations.

Graph simplification We drop degree-0 vertices, merge identical vertices, and split the graph into *independent* components [2, Corollary 2] when possible. We find an initial solution using the median heuristic [3, 5] and a local search that tries to move slices and optimally insert them [8, 4], and re-label all nodes accordingly to make memory accesses more efficient.

Fixed pairs We find all strongly fixed pairs and store them. For the exact track we also find practically fixed pairs. Instances for the parameterized track are simple enough that the overhead was not worth it. Also for each tail we search for new "tail-local" practically fixed pairs. In each state, we only try vertices $u \in T$ not fixed by another $v \in T$.

Gluing We use the greedy strategy of Lemma 11. Our implementation of Lemma 10 contained a bug, so we did not use this. (Also benefits seemed limited.)

Tail cache In each step, we search for the longest suffix of T that was seen before, and reuse (the lower bound on) its excess. We also cache the tail-local practically fixed pairs.

Optimal insert Instead of simply appending u to P, we insert it in the optimal position. Note that the implementation is tricky because it interacts in complicated ways with the caching of results for each tail.

References

1 Vida Dujmovic, Henning Fernau, and Michael Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. *Journal of Discrete Algorithms*, 6(2):313–323, June 2008. `doi:10.1016/j.jda.2006.12.008`.

2 Vida Dujmovic and Sue Whitesides. An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. *Algorithmica*, 40(1):15–31, April 2004. `doi:10.1007/s00453-004-1093-2`.

3 P. Eades and N.C. Wormald. *The Median Heuristic for Drawing 2-layered Networks*. Technical report. University of Queensland, Department of Computer Science, 1986.

4 Peter Eades and David Kelly. Heuristics for reducing crossings in 2-layered networks. *Ars Combinatoria*, 21(A):89–98, 1986.

5 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, April 1994. `doi:10.1007/bf01187020`.

6 Ragnar Groot Koerkamp and Mees de Vries. OCMu64. Software (visited on 2024-11-28). URL: `https://github.com/mjdv/ocmu64`, `doi:10.4230/artifacts.22525`.

7 Ragnar Groot Koerkamp and Mees de Vries. OCMu64. Software (visited on 2024-11-28). doi:10.5281/zenodo.11671980.

8 Erkki Mäkinen. Experiments on drawing 2-level hierarchical graphs. *International Journal of Computer Mathematics*, 36(3-4):175–181, January 1990. doi:10.1080/00207169008803921.