OCMu64: a solver for One-sided Crossing Minimization

Ragnar Groot Koerkamp
ETH Zurich, Switzerland

Mees de Vries

Abstract

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

We prove a number of novel reductions, and apply these in our solver OCMu64 that uses branch-and-bound on the order of the vertices of B.

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

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

Supplementary Material *Software (Source Code)*: `https://github.com/mjdv/ocmu64`

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 $(0, i)$ and $(1, j)$ for components A and B respectively. The order of A is fixed, and the goal is the find an order of B that minimizes the number of crossings when edges are drawn as straight lines.

We introduce some new reductions and overview our algorithm. Proofs are brief or omitted due to lack of space.

2 Definitions

Since the order of A is fixed, we set $A = \{0, \dots, [A]\}$ (where $[n] = \{0, \dots, n-1\}$) and draw the vertices in this order. Let $u, v \in B$ be two vertices in B. Then $N(u), N(v) \subseteq A$ are their sets of neighbours. We write $c(u, v) = \sum_{a\in N(u)} \sum_{b\in N(v)} [a > b]$ for the number of crossings between edges from 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 $u \preceq v$ when there exist minimal solutions where u comes before v, and we call such (u, v) *dominating pairs*. We write $u \sim v$ when there is a minimal solution where u and v are consecutive, in that order.

The following result is well known.

▶ **Observation 1.** *When all neighbours of* u *come before all neighbours of* v*, then* u *goes before* v*. I.e., when* $\max N(u) \le \min N(v)$*, then* $u \preceq v$*. When additionally* $\min N(u) < \max N(v)$*, then* $u \prec v$*.*

3 Methods

3.1 Reductions

Dominating pairs We give a much stronger version of Observation 1. In fact, this lemma is as strong as can be when only considering $N(u)$ and $N(v)$.

▶ **Lemma 2** (Strong dominating pair). *Write $n := |N(u)|$, $m := |N(v)|$, and consider both sets as sorted lists. When for all $i \in [n]$ we have $N(u)_i \leq N(v)_{\lfloor i \cdot m/n \rfloor}$, then $u \preceq v$. When there is an i such that the inequality is strict, then $u \prec v$.*

Proof. Consider for each $u \in B$ the function $w_u : A \to \mathbb{R}_{\geq 0}$ that assigns *weight* 1 to neighbours of u and weight 0 otherwise. Let $W_u = \sum_{a \in A} w_u(A)$ be the total weight (degree) of u. Then for any u and v, $c(u, v) = \sum_{a,b \in A, a>b} w_u(a) \cdot w_v(b)$. We can 'shift' some weight δ of w_u to the right from a_1 to $a_2 > a_1$ by decreasing $w_u(a_1) \geq \delta$ by δ and increasing $w_u(a_2)$ by δ. This increases the value of $c(u, v)$ by $\delta \cdot \sum_{a_1 < b \leq a_2} w_b(v) \geq 0$.

Consider *scaled* functions $\overline{w}_u = w_u/W_u$ and $\overline{w}_v = w_v/W_v$. We have $c(u, v) = W_u \cdot W_v \cdot c(\overline{u}, \overline{v})$ and $c(v, u) = W_u \cdot W_v \cdot c(\overline{v}, \overline{u})$, so that the sign of $r(u, v) = W_u \cdot W_v \cdot r(\overline{w}_u, \overline{w}_v)$ is independent of scaling. The condition in the lemma holds if and only if $\overline{w}_u$ can be transformed into $\overline{w}_v$ by shifting weight to the right.

Now suppose there is an optimal solution where v occurs before x: $(\dots, v, x_1, \dots, x_k, u, \dots)$. We set $X = (x_1, \dots, x_k)$ and write $w_X = \sum_{i=1}^{k} w_i$ for the combined weight of the neighbours of the x_i. Since the solution is optimal, we have $r(v, X) \leq 0$ and $r(X, u) \leq 0$, and hence $r(\overline{v}, X) \leq 0$ and $r(X, \overline{u}) \leq 0$. Since we can go from $\overline{w}_u$ to $\overline{w}_v$ by shifting weight to the right, we have $r(\overline{u}, X) \leq r(\overline{v}, X)$. We get $0 \leq -r(X, \overline{u}) = r(\overline{u}, X) \leq r(\overline{v}, X) \leq 0$. Thus, all must equal 0, and (v, X, u) is as good as (v, u, X). By a similar argument we can swap u and v to obtain (u, v, X). We conclude $u \preceq v$. When one of the inequalities in the condition is strict that implies $c(u, v) < c(v, u)$ and we have $u \prec v$. ◀

▶ Remark 3. When the condition in Lemma 2 does not hold, there exists a multiset $\mathcal{X} \subseteq A$ such that $v, \mathcal{X}, u$ is the optimal order.

Although such a set $\mathcal{X}$ may exist in theory, it does not have to exist in practice, motivating the following definition.

▶ **Lemma 4** (Practical dominating pair). *Suppose $r(u, v) \leq 0$. A* blocking *set $X \subseteq B - \{u, v\}$ is a set such that $c(v, X, u)$ is stricly better than both $c(u, v, X)$ and $c(X, u, v)$. If there is no blocking set, then $u \preceq v$.*

In practice, such a set X can be found, if one exists, using knapsack: 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 $\leq (r(u, v), r(u, v))$.

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 finding such sets much harder.

Gluing We now turn our attention to *gluing*, i.e., proving that two vertices u and v always go right next to each other. We start with a simple case of *gluing to the front.*

▶ **Lemma 5** (Greedy). *When $r(u, x) \leq 0$ for all $x \in B$, there is a solution that starts with u.*

▶ Remark 6. Let u and v satisfy $r(u, v) \leq 0$. When $N(u) = N(v)$ or more generally $\overline{w}_u = \overline{w}_v$, we can glue u and v: $u \sim v$. Otherwise, there is a multiset $\mathcal{X} \subseteq A$ such that (u, X, v) is better than (u, v, X) and (X, u, v).

This means that there is no 'strong gluing'.

▶ **Lemma 7** (Practical gluing). *Let u and v satisfy $r(u,v) \leq 0$ and $u \preceq v$. A subset $X \subseteq B - \{u,v\}$ is* blocking *when $c(u,X,v)$ is strictly better than $c(u,v,X)$ and $c(X,u,v)$. If there is no blocking set, then $u \sim v$.*

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 $\leq (0,0)$.

▶ Remark 8 (Tail variants). Our branch-and-bound method fixes vertices of the solution from left to right. That means that at each step Lemmas 4 and 7 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 order 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 $S_0 = \sum_{u,v} \min(c(u,v), c(v,u))$ 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 above 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 when possible. We find an initial solution using local search that tries to move slices and optimally insert them, and re-label all nodes accordingly to make memory accesses more efficient.

Dominating pairs We find all dominating pairs and store them. For the exact track we also find practical dominating 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' practical dominating pairs. In each state, we only try vertices $u \in T$ not dominated by another $v \in T$.

Gluing We use the greedy strategy Lemma 5. Our implementation of Lemma 7 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 has been seen before, and reuse (the lower bound on) its excess. We also cache the tail-local practical dominating pairs.

Optimal insert Instead of simply appending u to the end of P, we insert it in the optimal position.[1]

4 Discussion

`OCMu64` is a simple branch-and-bound method for solving one-sided crossing minimization that does not depend on advanced ILP or SAT solvers.

This also seems to be its biggest limitation: while LP methods can use the dual-space, `OCMu64` only considers the primal space to find lower bounds on subproblems.

[1] The idea is simple, but the implementation is tricky because it interacts in complicated ways with the caching of results for each tail.