# PACE Solver Description: CRGone

**Alexander Dobler**
TU Wien, Austria

## Abstract

We describe *CRGone*, our solver for the exact and parameterized track of the Pace Challenge 2024. It solves the problem of one-sided crossing minimization, is based on an integer linear programming (ILP) formulation with additional reduction rules, and is implemented in `C++` using the ILP solver `SCIP` with `Soplex`.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms; Human-centered computing → Graph drawings; Mathematics of computing → Permutations and combinations

**Keywords and phrases** Pace Challenge 2024, One-Layer Crossing Minimization, Exact Algorithm

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2024.29

**Supplementary Material**
*Software (Source Code)*: `https://zenodo.org/doi/10.5281/zenodo.11634869`

**Funding** *Alexander Dobler*: Supported by the Vienna Science and Technology Fund (WWTF) under grant 10.47379/ICT19035.

## 1 Introduction

One-sided crossing minimization (OSCM) is a problem from layered graph drawing and was first introduced by Sugiyama et al. [11]. The input is a bipartite graph $G = (V_t \dot\cup V_b, E)$, $E \subseteq V_t \times V_b$, with a fixed order $\pi_t$ of $V_t$. The question is to find an ordering $\pi_b$ of $V_b$, that minimizes the number of edge crossings when $G$ is drawn straight-line such that $V_t$ and $V_b$ are drawn on two respective horizontal lines $\ell_t$ and $\ell_b$ ordered according to $\pi_t$ and $\pi_b$, respectively. It is a purely combinatorial problem as two edges $(a, b), (c, d) \in V_t \times V_b$ cross if and only if

- $a \prec_{\pi_t} c$ and $d \prec_{\pi_b} b$, or
- $c \prec_{\pi_t} a$ and $b \prec_{\pi_b} d$

where $x \prec_\pi y$ means that $x$ comes before $y$ in the order $\pi$. The problem is NP-hard [4], heuristics [11, 4], and fixed-parameter algorithms with the natural parameter [3, 2, 8] are available. It has also been extensively studied with regard to integer linear programming [7].

In Section 2 we give some definitions and in Section 3 we describe our solver.

## 2 Definitions and Problem Insights

We assume that the input graph is a multigraph as some of our modifications introduce multiedges. For a set $X$, let $\binom{X}{i}$ be all the subsets of $X$ of size $i$. For a vertex $u$, let $N(u)$ be its adjacent vertices, and let $E(u)$ be its incident edges. Given $u \in V_b$, we define $s(u)$ as the open interval $(a, b)$ where $a$ is the minimum index of a neighbour of $u$ in $\pi_t$, and $b$ is the maximum index. For $u, v \in V_b$ ($u \neq v$), let $c(u, v)$ be the number of crossings between edge pairs in the set $E(u) \times E(v)$ when $u$ is placed before $v$ in $\pi_b$. We have that $\sum_{\{u,v\} \in \binom{V_b}{2}} \min(c(u, v), c(v, u))$ is a lower bound for the number of crossings and $\sum_{\{u,v\} \in \binom{V_b}{2}} \max(c(u, v), c(v, u))$ is an upper bound. We define $c_r(u, v) = c(u, v) - \min(c(u, v), c(v, u))$.

For an instance $G$ of OSCM let $G_d$ be the directed multi-graph with $V_b$ as vertex set such that for $\{u,v\} \in \binom{V_b}{2}$, there are $c(u,v)$ arcs from $v$ to $u$ and there are $c(v,u)$ arcs from $u$ to $v$. It is known that OSCM is equivalent to finding a minimum feedback arc set in $G_d$; a topological order of $G_d$ after removal of the minimum feedback arc set of size $k$ corresponds to an order of $\pi_b$ with $k$ crossings.

# 3 Solver description

We describe now our solver. It starts applying several reduction rules and decomposition rules which split the instance into multiple smaller instances. Then an integer linear programming formulation is applied that was optimized for sparse graphs.

## 3.1 Reduction rules

Here, we describe reduction and decomposition rules and how they were implemented and applied. The first rule is applied during preprocessing and is due to twins in $V_b$, which can be contracted.

▶ **Reduction Rule 1.** *Let $X \subseteq V_b$ maximal such that $|X| > 1$ and $\forall u, v \in X : N(u) = N(v)$. Contract $X$ into a single vertex that has multiedges of multiplicity $|X|$.*

We find such sets $X$ using a trie. The values $c(u,v)$ for the reduced instance are only computed afterward. The next is due to the formulation as feedback arc set problem.

▶ **Decomposition Rule 2** ([9])**.** *Let $T = (G_1, G_2, \ldots, G_p)$ be a topological order of the strongly connected components of $G_d$. Then split the instance into $G_1, G_2, \ldots, G_p$, whose individual solutions are then concatenated according to the topological order.*

We also implemented decomposition rules based on biconnected components of $G_d$ [10], which almost never applied to the input instances, so it was not included in the final submission.

The next reduction rule fixes the relative order of pairs of vertices in $V_b$.

▶ **Reduction Rule 3** ([3])**.** *If there exist $u, v \in V_b$ with $c(u,v) = 0$, fix $u \prec_{\pi_2} v$.*

The last reduction rule is more complicated and is related to modular decompositions of two-structures [6].

▶ **Decomposition Rule 4.** *Let $X \subsetneq V_b$, $|X| > 1$, such that*

$$\forall u, v \in X \forall w \in V_b \setminus X : c_r(u,w) = c_r(v,w) \wedge c_r(w,u) = c_r(w,v).$$

*Then compute an optimal order $\pi_1$ of $X$ for $G[V_t \cup X]$. Let $G_c$ be the graph obtained from $G$ by contracting $X$ into a single vertex $x$. Compute an optimal order $\pi_2$ of $(V_b \setminus X) \cup \{x\}$ in the reduced instance $G_c$. By replacing in $\pi_2$ the contracted vertex $x$ by $\pi_1$, we obtain an optimal solution.*

The sets $X$ above not containing a randomly chosen $y \in V_b$ are computed using a partition refinement algorithm as described in [6]. The above decomposition rules are applied recursively with decreasing priority, i.e., Decomposition Rule 2 has the highest priority, Reduction Rule 3 is only applied afterward to the decomposed parts, Decomposition Rule 4 has the lowest priority. We also implemented the reduction rules, which fix relative orders of vertex-pairs based on 2/1-structures from [2] with the same priority as Reduction Rule 3.

## 3.2 Integer linear program

If no decomposition and reduction rules are applicable, we employ an integer linear program. The formulation is as in [7]. Assume a total order $<$ on $V_b$. For each pair $u, v \in V_b$, $u < v$ we have a binary *ordering variable* $x_{u,v}$ which is 1 if and only if $u \prec_{\pi_b} v$. The formulation is as follows.

$$\min \quad \sum_{u,v \in V_b, u<v} (c(v,u) + x_{u,v}(c(u,v) - c(v,u))) \tag{ILP}$$

$$0 \leq x_{u,v} + x_{v,w} - x_{u,w} \leq 1 \qquad u, v, w \in V_b, u < v < w \tag{TRANS}$$

$$x_{u,v} \in \{0,1\} \qquad u, v \in V_b, u < v \tag{BIN}$$

**Adaptations.** Due to Reduction Rule 3 and the reduction rules from [2] we know the relative order of specific pairs of vertices from $V_b$. This means that for some ordering variables $x_{u,v}$ we already know that they are 1 or 0. We remove those variables from the model and replace them by the corresponding constant in the above model. Resulting constraints which are satisfied regardless of variable assignment are removed.

Next, the (TRANS)-constraints are separated using a branch and cut approach. This involves first categorizing the (TRANS) constraints based on the values of $s(u), s(v), s(w)$ for the vertices $u, v, w$ in each constraint.

- If $s(u) \cap s(v) \cap s(w) \neq \emptyset$, then it is a *type-1* constraint.
- If a constraint is type-1 and additionally, there are two pairs $x, y$ and $p, q$ among the triple $u, v, w$ such that $c(x, y) \neq c(y, x)$ and $c(p, q) \neq c(q, p)$, then it is *weak-type-1*.
- If a constraint is not type-1, it is a *type-2* constraint.

The idea is that type-1 constraints can be enumerated quickly without storing them by using a sweep-line over the sorted interval borders of $s(u)$ for all $u \in V_b$. We enumerate type-2 constraints by saving for each vertex $u \in V_b$ the set $S(u)$ of vertices $v \in V_b$ with $s(u) \cap s(v) \neq \emptyset$. Then the type-2 constraints (which stay after applying reduction rules) can be found by enumerating pairs in $S(u)$ for each $u \in V_b$. The solver is initialized without any (TRANS)-constraints. First, only violated weak-type-1 constraints are separated. Once, there is a separation round where no violated weak-type-1 constraint can be separated, type-1 constraints are separated from now on. Lastly, type-2 constraints are only separated in a separation round if there are no type-1 constraints that can be separated.

Lastly, we implemented a heuristic that exploits fractional solutions. To this end, we start with a feasible solution $\hat{\pi_b}$ conforming to the reduction rules. Then we compute values $p(u)$ for all $u \in V_b$: for each ordering variable $x_{u,v}$ let $y_{u,v}$ be the rounded value in the fractional solution. We add 1 to $p(v)$ if $y_{u,v} = 1$, otherwise, we add 1 to $p(u)$. Lastly, for all $u \in V_b$ we add to $p(u)$ the number of $v \in V_b$ that are fixed before $u$ according to the reduction rules. The heuristic computes an ordering starting with $\hat{\pi_b}$ and swapping two adjacent vertices $u, v$ in $\hat{\pi_b}$ ($u$ before $v$) where $p(u) > p(v)$ and $u$ does not have to be before $v$ according to the reduction rules. This is implemented in a bubble-sort-like algorithm. The worst-case runtime is equal to the number of ordering variable in the reduced model.

The above is implemented in `C++17` using `SCIP` version 9 [1] with `Soplex` version 7 [5]. Parameters are chosen such that the solver remains in the root of the branch and bound tree as long as possible.

## References


1 Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, et al. The SCIP optimization suite 9.0, 2024. `arXiv:2402.17702`.

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

3 Vida Dujmovic and Sue Whitesides. An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. *Algorithmica*, 40(1):15–31, 2004. `doi:10.1007/S00453-004-1093-2`.

4 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. `doi:10.1007/BF01187020`.

5 Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, et al. The SCIP optimization suite 7.0, 2020. URL: `https://opus4.kobv.de/opus4-zib/files/7802/scipopt-70.pdf`.

6 Michel Habib and Christophe Paul. A survey of the algorithmic aspects of modular decomposition. *Comput. Sci. Rev.*, 4(1):41–59, 2010. `doi:10.1016/J.COSREV.2010.01.001`.

7 Michael Jünger and Petra Mutzel. Exact and heuristic algorithms for 2-layer straightline crossing minimization. In Franz-Josef Brandenburg, editor, *Proc. Symposium on Graph Drawing and Network Visualizations (GD'95)*, volume 1027 of *LNCS*, pages 337–348. Springer, 1995. `doi:10.1007/BFB0021817`.

8 Yasuaki Kobayashi and Hisao Tamaki. A fast and simple subexponential fixed parameter algorithm for one-sided crossing minimization. *Algorithmica*, 72(3):778–790, 2015. `doi:10.1007/S00453-014-9872-X`.

9 Sungju Park and Sheldon B Akers. An efficient method for finding a minimal feedback arc set in directed graphs. In *Proc. International Symposium on Circuits and Systems (ISCAS'92)*, volume 4, pages 1863–1866. IEEE, 1992.

10 Hermann Stamm. On feedback problems in planar digraphs. In Rolf H. Möhring, editor, *Proc. Graph-Theoretic Concepts in Computer Science (WG'90)*, volume 484 of *LNCS*, pages 79–89. Springer, 1990. `doi:10.1007/3-540-53832-1_33`.

11 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.*, 11(2):109–125, 1981. `doi:10.1109/TSMC.1981.4308636`.