MPPEG

Michael Jünger
University of Cologne, Germany

Paul Jünger
University of Bonn, Germany

Petra Mutzel
University of Bonn, Germany

Gerhard Reinelt
Heidelberg University, Germany

Abstract

This is a short description of our solver `oscm` submitted by our team MPPEG to the PACE 2024 challenge both for the exact track and the parameterized track, available at `https://doi.org/10.5281/zenodo.11546972` [9] and `https://github.com/pauljngr/PACE2024` [8].

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Theory of computation → Mathematical optimization; Theory of computation → Parameterized complexity and exact algorithms; Human-centered computing → Graph drawings

Keywords and phrases Combinatorial Optimization, Linear Ordering, Crossing Minimization, Branch and Cut, Algorithm Engineering

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Method

We apply the approach to the one-sided crossing minimization problem presented in [10]. This article is surveyed by Patrick Healy and Nikola S. Nikolov in Chapter 13.5 of the Handbook of Graph Drawing and Visualization [7] that is recommended on the PACE 2024 web page. The method consists of a transformation of a one-sided crossing minimization instance to an instance of the linear ordering problem that is solved by branch&cut as introduced in [4] and [5]. We also use problem decomposition and reduction techniques as well as a heuristic for finding a good initial solution. With the required brevity, we give a rough sketch of the major details.

The instances of the PACE 2024 challenge problem consist of a bipartite graph $G = (T\dot{\cup}B, E)$ and a fixed linear ordering $\pi_T = \langle t_1, t_2, \ldots, t_m\rangle$ of T ("the top nodes"). In the exact track and the parameterized track, the task is to find a linear ordering π_B of $B = \{b_1, b_2, \ldots, b_n\}$ ("the bottom nodes") such that the number of edge crossings in a straight-line drawing of G with T and B on two parallel lines, following their linear orderings, is provably minimum. The NP-hardness of this task has been shown in [2].

For a linear ordering π_B of B let

$$x_{ij} = \begin{cases} 1 & \text{if } b_i \text{ appears before } b_j \text{ in } \pi_B, \\ 0 & \text{otherwise.} \end{cases}$$

For $i, j \in \{1, 2, \ldots, n\}$ let $c_{ii} = 0$, and for $i \neq j$ let c_{ij} denote the number of crossings between the edges incident to b_i with the edges incident to b_j if b_i appears before b_j in π_B. Then the number of crossings induced by π_B is

$$\mathrm{cr}(\pi_B) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}.$$

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:4
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Since for any pair $b_i \neq b_j$ in B we have $x_{ji} = 1 - x_{ij}$, we can reduce the number of variables to $\binom{n}{2}$ and obtain

$$\mathrm{cr}(\pi_B) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}x_{ij} + c_{ji}(1 - x_{ij}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (c_{ij} - c_{ji})x_{ij} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ji}.$$

For $a_{ij} = c_{ij} - c_{ji}$ we solve the *linear ordering problem* as the following binary linear program, based on the complete digraph D with node set B.

$$\begin{array}{llll}
\text{(LO)} & \text{minimize} & \displaystyle\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{ij}x_{ij} & \\
& \text{subject to} & \displaystyle\sum_{\substack{(b_i,b_j)\in C:\\ i<j}} x_{ij} + \sum_{\substack{(b_i,b_j)\in C:\\ i>j}} (1 - x_{ji}) \leq |C| - 1 & \text{for all dicycles } C \text{ in } D \\
& & 0 \leq x_{ij} \leq 1 & \text{for } 1 \leq i < j \leq n \\
& & x_{ij} \text{ integral} & \text{for } 1 \leq i < j \leq n.
\end{array}$$

If z is the optimum value of (LO), $z + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ji}$ is the minimum number of crossings. Notice that the classical linear ordering formulation [4, 5] uses constraints for cycles of length three only. However, in our approach we also need longer cycles, since we remove some of the arcs as we shall describe in Section 2. The constraints of (LO) guarantee that the solutions correspond precisely to all permutations π_B of B. Furthermore, it can be shown that for complete digraphs the "3-cycle constraints" are necessary in any minimal description of the feasible solutions by linear inequalities, if the integrality conditions are dropped. The NP-hardness of the problem makes it unlikely that such a complete linear description can be found and exploited algorithmically. Further classes of inequalities with a number of members exponential in n that must be present in a complete linear description of the feasible set, are known, and some of them can be exploited algorithmically. Indeed, small Möbius-ladder constraints, the one shown in Figure 3 of [4], as well as the same in which all arcs are reversed, have been found useful in this crossing minimization context.

2 Algorithm and Implementation

When the integrality conditions in (LO) are dropped, we obtain a linear programming relaxation of (LO) which has been proven very useful in practical applications. The structure of our branch&cut algorithm `oscm` ("**o**ne-**s**ided **c**rossing **m**inimization") is similar to the one proposed in [4]. The algorithm starts with the trivial constraints $0 \leq x_{ij} \leq 1$ that are handled implicitly by the linear program solver, iteratively adds violated cycle and Möbius-ladder constraints, and deletes nonbinding constraints after a linear program has been solved, until the relaxation is solved. This requires a *separation* algorithm that, given the solution of some relaxation, is able to determine a violated inequality called *cutting plane*. If the optimum solution of the relaxation is integral, the algorithm stops, otherwise it is applied recursively to two subproblems in one of which a fractional x_{ij} is set to 1 and in the other set to 0. Thus, in the end, an optimum solution is found as the solution of some relaxation, along with a proof of optimality.

`oscm` makes use of the following observations, some of which stem from the literature in fixed-parameter algorithms for one-sided crossing minimization. Lemma 1 allows us to decompose the given instance. Within the components, we can fix and eliminate variables from (LO) by Lemma 2, and we can exclude variables x_{ij} with $a_{ij} = 0$ from (LO) by Lemma 3.

▶ **Lemma 1** (Decomposition). *For each node $v \in B$, we define the open interval $I_v =]l_v, r_v[$, where l_v is the position of the leftmost and r_v the position of the rightmost neighbor of v in π_T. The union of the intervals I_v induces a partition $B_1, B_2, \dots, B_k$ of B such that every $I_{B_i} = \bigcup_{v \in B_i} I_v$, $i = 1, \dots, k$, is an interval, and for any pair B_i, B_j the intervals I_{B_i} and I_{B_j} are disjoint. In every optimum π_B all the nodes of B_i appear before those of B_j if I_{B_i} is to the left of I_{B_j}.*

Indeed, 51 of the 100 exact-public instances have between 2 and 154 components.

▶ **Lemma 2** (Variable fixing [1, 11]). *If for any pair of nodes $b_i, b_j \in B$, we have $c_{ij} = 0$ and $c_{ji} > 0$, then every optimal solution of (LO) satisfies $x_{ij} = 1$, if $i < j$, or $x_{ji} = 0$, if $i > j$.*

▶ **Lemma 3** (Arbitrary ordering). *Let $\pi_B^{(p)}$ be a partial ordering induced by the variables x_{ij} with $a_{ij} \neq 0$, then there exist values $x_{ij} \in \{0, 1\}$ for $a_{ij} = 0$ defining a total ordering π_B of B with no effect to the objective function value. This assignment can be found by topologically sorting B with respect to $\pi_B^{(p)}$.*

This setup has the advantage that (sometimes considerably) smaller linear programs need to be solved, but, on the other hand, separation becomes more involved. In order to obtain an optimal partial ordering $\pi_B^{(p)}$ of B using the variables left in (LO), we need to include cycle constraints for larger cycles as already mentioned in Section 1.

For computational efficiency, `oscm` has a hierarchy of separation procedures. The first for 3-dicycles is based on depth first search. The second for dicycles of length at least 4 with integral weights is also based on depth first search. Violated dicycles are shortened via breadth first search, restricted to the cycle nodes, starting from back arcs of the preceding depth first search. The third applies shortest path techniques for separation of cycles containing fractional arcs as described for the related acyclic subdigraph problem in section 5 of [6]. First, the above separation procedures are applied on the graph containing only the arcs present in (LO). If all of the above do not find any violated inequalities, `oscm` extends the search to the fixed arcs. After separation, the linear program is resolved using the dual simplex method providing the same or a better lower bound on the minimum number of crossings. If the progress compared to the previous bound is small for a sequence of such lower bounds, `oscm` applies a heuristic for finding violated Möbius ladder inequalities, and if this does not lead to a significant improvement, the branch&cut phase is started.

Whenever a linear program has been solved, it is checked by topological sorting if the solution is the characteristic vector of a linear ordering. If not, a relaxed topological sorting procedure is applied in the pursuit of finding a better incumbent solution that provides an upper bound for the minimum number of crossings. `oscm` stops when the (integral) upper bound and the (possibly fractional) lower bound differ by less than 1, proving optimality.

For small instances, `oscm` applies a variant of the heuristic "Kernighan-Lin 2" of [13] for finding a decent initial solution before the optimization starts.

3 Performance

Our program `oscm` consists of roughly 3500 lines of C/C++ code. It makes use of the coin-or [12] Cbc library, version 2.10.7 [3].

We have measured the performance of `oscm` on one thread of an Intel Xeon "Sapphire Rapids" 2.10GHz. Applied to the exact-public instance set, `oscm` receives a 30 minute timeout for instance `92.gr`, but solves within about 51 minutes all remaining 99 instances with individual times ranging from less than 1 second to about 17 minutes. Applied to the cutwidth-public instance set, solving all 125 instances takes 21 seconds. See [9] and [8].

References

1. Vida Dujmović and Sue Whitesides. An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. *Algorithmica*, 40(1):15–31, 2004. URL: `https://doi.org/10.1007/s00453-004-1093-2`, `doi:10.1007/S00453-004-1093-2`.
2. Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. `doi:10.1007/BF01187020`.
3. John Forrest et al. coin-or/Cbc: Release 2.10.7, 2022. URL: `https://zenodo.org/records/5904374`.
4. Martin Grötschel, Michael Jünger, and Gerhard Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32(6):1195–1220, 1984. URL: `https://doi.org/10.1287/opre.32.6.1195`.
5. Martin Grötschel, Michael Jünger, and Gerhard Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33(1):43–60, 1985. `doi:10.1007/BF01582010`.
6. Martin Grötschel, Michael Jünger, and Gerhard Reinelt. On the acyclic subgraph polytope. *Mathematical Programming*, 33(1):28–42, 1985. `doi:10.1007/BF01582009`.
7. Patrick Healy and Nikola S. Nikolov. Hierarchical drawing algorithms. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, Discrete Mathematics and Its Applications, pages 409–453. CRC Press, 2013. URL: `https://api.semanticscholar.org/CorpusID:7736149`.
8. Michael Jünger, Paul Jünger, Petra Mutzel, and Gerhard Reinelt. OSCM, June 2024. URL: `https://github.com/pauljngr/PACE2024`, `doi:10.5281/zenodo.11546972`.
9. Michael Jünger, Paul Jünger, Petra Mutzel, and Gerhard Reinelt. PACE 2024 - MPPEG, June 2024. `doi:10.5281/zenodo.11546972`.
10. Michael Jünger and Petra Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Applications*, 1(1):1–25, 1997. `doi:10.7155/jgaa.00001`.
11. Yasuaki Kobayashi and Hisao Tamaki. A fast and simple subexponential fixed parameter algorithm for one-sided crossing minimization. *Algorithmica*, 72(3):778–790, 2015. URL: `https://doi.org/10.1007/s00453-014-9872-x`, `doi:10.1007/S00453-014-9872-X`.
12. R. Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003. `doi:10.1147/rd.471.0057`.
13. Rafael Martí and Gerhard Reinelt. *Exact and Heuristic Methods in Combinatorial Optimization - A Study on the Linear Ordering and the Maximum Diversity Problem*. Applied Mathematical Sciences. Springer Berlin, Heidelberg, 2022. URL: `https://doi.org/10.1007/978-3-662-64877-3`.