


PACE Solver Description: Crossy – An Exact Solver for One-Sided Crossing Minimization

Tobias Röhr ✉

Hasso Plattner Institute, University of Potsdam, Germany

Kirill Simonov ✉ 

Hasso Plattner Institute, University of Potsdam, Germany

Abstract

We describe Crossy, an exact solver for One-sided Crossing Minimization (OSCM) that ranked 5th in the Parameterized Algorithms and Computational Experiments (PACE) Challenge 2024 (Exact and Parameterized Track). Crossy applies a series of reductions and subsequently transforms the input into an instance of Weighted Directed Feedback Arc Set (WDFAS), which is then formulated in incremental MaxSAT. We use the recently introduced concept of User Propagators for CDCL SAT solvers in order to dynamically add cycle constraints.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases One-sided Crossing Minimization, Exact Algorithms, Graph Drawing, Incremental MaxSAT

Digital Object Identifier 10.4230/LIPIcs.IPEC.2024.30

Supplementary Material *Software (Source Code)*: <https://github.com/roehrt/crossy>

archived at `swh:1:dir:aa5e0abc800f7a51008548897226b54b38032404`

Software (Source Code): <https://doi.org/10.5281/zenodo.12082773>

1 Preliminaries

Given a bipartite graph $G = (A, B, E)$ and a linear ordering on the vertices of A , the One-sided Crossing Minimization problem asks for a linear ordering \prec on the vertices of B that minimizes the number of crossings of a straight-line drawing when placing the vertices in A and B on two parallel lines in the respective order. To enable some of our reduction rules, it is convenient to relax the problem and allow the input to be a multigraph.

For $u, v \in B$, define $c(u, v)$ to be the number of crossings between the edges incident to u and v when $u \prec v$. Moreover, we call $u \prec v$ the *natural order* of u and v if and only if $c(u, v) < c(v, u)$. Since in any solution either $u \prec v$ or $v \prec u$, we get a simple lower bound on the number of crossings: $\sum_{u, v \in B} \min(c(u, v), c(v, u))$.

The penalty graph of an OSCM-instance is a directed graph on the vertices of B . In order to penalize pairs of vertices that do not appear in their natural order, we add an arc $u \rightarrow v$ carrying weight $c(u, v) - c(v, u)$ for any pair $u, v \in B$ with $c(u, v) > c(v, u)$. Note that the weight of a Minimum Weight Feedback Arc Set in the penalty graph equals the minimum number of crossings in the corresponding OSCM-instance above its lower bound [8].

We say that we *commit* $u \prec v$ if we only look for solutions where u appears before v . To model this knowledge in the penalty graph, we insert an arc $u \rightarrow v$ with infinite weight in this case.



© Tobias Röhr and Kirill Simonov;

licensed under Creative Commons License CC-BY 4.0

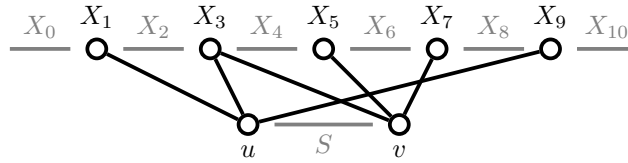
19th International Symposium on Parameterized and Exact Computation (IPEC 2024).

Editors: Édouard Bonnet and Paweł Rzażewski; Article No. 30; pp. 30:1–30:4

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** A configuration with $u \prec v$. S is the set of vertices between u and v . The sets X_i partition the neighborhood of S . The figure is adapted from [3].

2 Reduction rules

After merging twins in B and removing isolated vertices, we apply two sets of reduction rules: one specific to OSCM and the other consisting of general-purpose rules for the Weighted Directed Feedback Arc Set problem.

2.1 Rules for OSCM

We utilize two well-known rules for OSCM that we call Planar Ordering and Transitivity, and introduce a new rule, Dominance.

Planar Ordering If there is a pair of vertices $u, v \in B$ such that $c(u, v) = 0$, commit $u \prec v$.

Transitivity If $u \prec v$ and $v \prec w$, commit $u \prec w$.

The Planar Ordering reduction rule is due to Dujmovic and Whitesides [4]; see their work for the proof of correctness. The Transitivity rule is straightforward.

Before we formally define Dominance, consider the following argument showing that the natural order $u \prec v$ is optimal, in a certain case. Assume by contradiction that $u \prec v$ is not optimal, and consider an optimal ordering where v appears before u instead. Let S be the set of vertices between v and u in this ordering, i.e., $S = \{w \in B \mid v \prec w \prec u\}$.

In order for $u \prec v$ not to be optimal, the number of crossings caused by $v \prec S \prec u$ must be strictly less than the number of crossings caused by all configurations with $u \prec v$, namely: $u \prec S \prec v$, $u \prec v \prec S$, and $S \prec u \prec v$. If we can derive a contradiction for each possible set S , we have proven that $u \prec v$ is optimal and are able to commit $u \prec v$.

Inspired by the tabular analysis technique of Dujmovic, Fernau and Kaufmann [3], let us categorize the neighbors of S into disjoint sets based on their relative position to the neighbors of u and v , so that the neighbors in the same set are effectively indistinguishable with respect to the condition above (see Figure 1).

By introducing variables describing the cardinality of those sets, we can express the number of crossings for each configuration as a linear combination of these variables. Let $L(x, y, z)$ denote this linear combination for some configuration $x \prec y \prec z$. Now we can derive a system of linear inequalities that must hold for $u \prec v$ not to be optimal:

$$L(v, S, u) < L(S, u, v)$$

$$L(v, S, u) < L(u, S, v)$$

$$L(v, S, u) < L(u, v, S).$$

Additionally, each variable can also be bounded by the number of edges outgoing from each partition. Finally, we can use an LP solver to check the feasibility of this system of inequalities, leading us to the following rule:

Dominance If there is a pair of vertices $u, v \in B$ such that the LP derived from the above analysis is infeasible, commit $u \prec v$.

Thus, the Planar Ordering rule is a special case of the Dominance rule.

Although Dominance runs in polynomial time, it is computationally expensive as it requires solving a linear program for each vertex pair. To mitigate this, we use a weaker rule that removes upper bound constraints and checks only pairwise inequality feasibility.

This can be done in linear time, resulting in $\mathcal{O}(|B| \cdot |E|)$ for all OSCM-reductions.

2.2 Rules for WDFAS

Crossy proceeds to apply very general rules for the Weighted Directed Feedback Arc Set problem on the penalty graph.

Strongly Connected Components Find a solution for each strongly connected component, then combine the orderings following a topological ordering of the condensation graph.

Minimum Cut For each arc $u \rightarrow v$, commit $u \prec v$, if the weighted minimum cut separating v from u does not exceed the weight of $u \rightarrow v$.

Although the Minimum Cut rule can commit some pairs, our experiments show that the additional computational cost is not justified. We therefore disable it in our implementation, resulting in the overall running time of $\mathcal{O}(|B| \cdot |E|)$ for all preprocessing steps.

3 Incremental MaxSAT formulation

Building on the work of the winning team of the PACE Challenge 2022 [6], we formulate the Weighted Directed Feedback Arc Set problem as incremental MaxSAT problem, aiming to hit all cycles in the penalty graph.

3.1 Encoding

For each arc $u \rightarrow v$ in the penalty graph, we introduce a variable $x_{u \rightarrow v}$ representing the arc's inclusion in the solution. Each variable is assigned the negated weight of its corresponding arc. If an arc has infinite weight, the variable is set to false. To encode a cycle constraint, we add a disjunction of the variables corresponding to the arcs in that cycle.

Crossy starts by adding short cycles of length at most 4 to the MaxSAT instance explicitly. All longer cycles are added dynamically later by the user propagator.

3.2 User Propagator

We modify UWMaxSAT [7] to allow us to connect a user propagator to its underlying SAT solver, CaDiCal [1]. The recently introduced IPASIR-UP interface [5] enables us to connect user-defined propagators to CaDiCal without modifying the solver itself.

Our user propagator employs the concept of Cycle Propagation as introduced by Kiesel and Schidler [6]. Assigning a negative literal $x_{u \rightarrow v}$ indicates that the arc $u \rightarrow v$ is not included in the solution and, therefore, remains in the graph. Following the SAT solver's decision, our user propagator detects if the current assignment would close a cycle. If a cycle is detected, we add the corresponding cycle constraint to our MaxSAT instance, effectively pruning the current branch.

We tackle incremental cycle detection with rollbacks by eagerly maintaining the depth of each vertex. Upon each arc insertion, we recursively update the depths of affected vertices and check for any newly formed cycles.

As many arcs have infinite weight and thus always remain in the graph, we first compute the transitive reduction of this subgraph to reduce the workload for our cycle detection.

4 Discussion

While its performance secured Crossy the 5th place in PACE 2024, there are some challenges that arise due to its MaxSAT-based cycle-hitting formulation.

The penalty graph of an OSCM instance is known to be very dense, and our cycle-hitting formulation can result in a large, potentially exponential, number of constraints, even with the use of a user propagator. An alternative approach, employed by the top 3 teams, uses transitivity constraints for each triplet of vertices in B , which appears to perform better.

Moreover, OSCM allows for the application of various effective primal heuristics that can enhance ILP-based approaches but are not applicable to MaxSAT-based solvers like Crossy.

Finally, the performance of UWMaxSAT in MaxSAT competitions benefits from running SCIP [2] beforehand. This dependency poses a significant drawback for Crossy, as SCIP cannot take advantage of the user propagator.

References

- 1 Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froyleys, and Florian Pollitt. Cadical 2.0. In Arie Gurfinkel and Vijay Ganesh, editors, *Computer Aided Verification – 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24–27, 2024, Proceedings, Part I*, volume 14681 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 2024. doi:10.1007/978-3-031-65627-9_7.
- 2 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. *arXiv preprint*, 2024. arXiv:2402.17702.
- 3 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.
- 4 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.
- 5 Katalin Fazekas, Aina Niemetz, Mathias Preiner, Markus Kirchweger, Stefan Szeider, and Armin Biere. IPASIR-UP: user propagators for CDCL. In Meena Mahajan and Friedrich Slivovsky, editors, *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4–8, 2023, Alghero, Italy*, volume 271 of *LIPICs*, pages 8:1–8:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SAT.2023.8.
- 6 Rafael Kiesel and André Schidler. PACE solver description: DAGer – Cutting out cycles with maxsat. In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, September 7–9, 2022, Potsdam, Germany*, volume 249 of *LIPICs*, pages 32:1–32:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.IPEC.2022.32.
- 7 Marek Piotrów. UWMaxSat: Efficient solver for maxsat and pseudo-boolean problems. In *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9–11, 2020*, pages 132–136. IEEE, 2020. doi:10.1109/ICTAI50040.2020.00031.
- 8 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiro 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.