Sherby solver description for the parameterized and exact tracks of PACE 2024

Manuel Lafond
Department of Computer Science, University of Sherbrooke, 2500 Boulevard de l'Université, Sherbrooke, QC J1K 2R1

Alitzel López Sánchez
Department of Computer Science, University of Sherbrooke, 2500 Boulevard de l'Université, Sherbrooke, QC J1K 2R1

Bertrand Marchand
Department of Computer Science, University of Sherbrooke, 2500 Boulevard de l'Université, Sherbrooke, QC J1K 2R1

Abstract

We have submitted solvers to the exact and parameterized tracks of PACE 2024. They are based on the dynamic programming (DP) algorithm of [8], and the SAT-based method of [7] for directed feedback vertex set. As the run-time of our implementation of the DP approach of [8] is roughly known in advance given an instance (depending on its "interval structure"), we use an estimation of this run-time to decide when to use one approach or the other. When it is low enough, we apply [8]. When it is too high, we apply the SAT-based approach. We also implemented two reduction rules: one based on twins (vertices with the same sets of neighbors), and one based on the computation of strongly connected components of an associated graph.

2012 ACM Subject Classification

Keywords and phrases One-sided crossing minimization - dynamic programming - weighted directed feedback arc sat

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Related Version (solver DOI) https://zenodo.org/doi/10.5281/zenodo.12193623

Supplementary Material `https://github.com/bmarchand/sherby-pace-2024`

1 Problem definitions and processing rules

Let us first recall the definition of the problem:

> One-Sided Crossing Minimization
> **Input:** bipartite graph G with parts A and B, π_A order on A
> **Output:** An order π on B minimizing the number of *crossings*, i.e. pairs of edges u, v, w, x verifying $u \leq_{\pi_A} w$ and $x \leq_\pi v$

Note that whether two edges cross only depends on the relative position of their B end-points, as the two other end-points are fixed. It is therefore common (see for instance [4, 3, 8]) to define *crossing values* $c(u, v)$ for u and v in B, as the number of edge crossings involving an edge adjacent to u and another to v, when u is placed before v. With that definition, the number of crossings incurred by an order π on B is:

$$\mathsf{cr}(\pi) = \sum_{u \leq_\pi v} c(u, v)$$

For a set $S \subseteq B$, we write $c(S, u) = \sum_{v \in S} c(v, u)$, and define $c(u, S)$ similarly.

© Jane Open Access and Joan R. Public;
licensed under Creative Commons License CC-BY 4.0

1.1 Twin rule

Let us know give our first reduction rule, based on twins. Two vertices u and v are said *adjacent* in an order if u comes right before v or v right before u in that order.

▶ **Rule 1.** *If $N(u) = N(v)$, for u, v vertices of B, then there is an optimal ordering π in which u and v are adjacent.*

Proof. Note that since u and v have exactly, the same neighbors, they play perfectly symmetrical roles in the problem, and $c(u, v) = c(v, u)$. Consider G, π_A an instance of OSCM, and π an optimal ordering. Suppose u and v are not adjacent in π (say with u first), and let S the set of vertices after u and before v in π.

We consider the orders π' and π'', identical to π except that u has been moved right after v in π' and v has been moved right before u in π''. We argue that one of π' or π'' is also optimal for G, π_A.

Suppose indeed the opposite. We would then have $\mathsf{cr}(\pi) < \mathsf{cr}(\pi')$ and $\mathsf{cr}(\pi) < \mathsf{cr}(\pi'')$. However, we also have

$$\mathsf{cr}(\pi') = \mathsf{cr}(\pi) - c(u, S) + c(S, u)$$

and

$$\mathsf{cr}(\pi'') = \mathsf{cr}(\pi) - c(S, v) + c(v, S)$$

The non-optimality of π' and π'' would therefore imply $c(S, u) > c(u, S)$ and $c(v, S) > c(S, v)$.

However, $c(u, S) = c(v, S)$. To see this, consider the positions of u, v and S in π''. A crossing between an edge u, x ($x \in A$) and an edge y, s for $s \in S$ also crosses edge v, x, as u and v have the same neighbors. But likewise, a crossing between an edge v, x and an edge y, s for $s \in S$ implies a crossing between u, x and y, s. Overall $c(u, S) = c(v, S)$. Likewise, $c(S, u) = c(S, v)$.

Therefore, if π' and π'' are both not optimal $c(S, u) > c(u, S) = c(v, S) > c(S, v) = c(S, u)$, which is a contradiction. ◀

Rule 1 implies that one may keep only one vertex out of the two twins, solve the instance, and re-insert the deleted vertex next to its twin afterwards. Note that the crossing values need to be updated accordingly: if u is kept, $c(w, u)$ for $w \neq v$ needs to be updated to $c(w, u) + c(w, v)$, and likewise for $c(v, w)$.

1.2 Reduction to weighted directed feedback arc set and SCC rule

In the literature, one may find two different reductions of OSCM to Weighted Directed Feedback Arc Set (wDFAS). In both of them, the vertex set of the directed graph is B. The difference is in the definition of the edge sets. In the first approach, an edge $u \to v$ is added to the graph if and only if $c(u, v) > 0$. This construction is for instance used in [5], where the fact that it yields (if vertices for which $c(u, v) = c(v, u) = 0$, i.e. degree-1 twins, are pre-processed) a "quasi-tournament" (for any pair u, v of vertices there is at least one edge between them) is exploited algorithmically.

In our solver, we have used another definition, where $u \to v$ if and only if $c(u, v) < c(v, u)$. In addition, $u \to v$ is given a weight $c(v, u) - c(u, v)$. This construction is known as the "penalty graph" [2]. For an instance G, π_A of OSCM, we call H_G the resulting graph.

The equivalence between solving OSCM and wDFAS on H_G is shown below. Given an order π on B (which is therefore an order on the vertices of H_G), write $\mathsf{w\text{-}dfas}(\pi)$ for the sum of the weights of the edges $u \to v$ such that $v \leq_\pi u$.

▶ **Proposition 1.** *Let G, π_A be an instance of OSCM, and π and order on B:*

$$\mathit{cr}(\pi) = \sum_{u \leq_\pi v} \min(c(u,v), c(v,u)) + \text{w-dfas}(\pi)$$

Proof.

$$\begin{aligned}
\mathsf{cr}(\pi) &= \sum_{u \leq_\pi v} c(u,v) \\
&= \sum_{u \leq_\pi v} \min(c(u,v), c(v,u)) + \sum_{\substack{v \to u \\ u \leq_\pi v}} c(v,u) - c(u,v) \\
&= \sum_{u \leq_\pi v} \min(c(u,v), c(v,u)) + \text{w-dfas}(\pi)
\end{aligned}$$

◀

As $\sum_{u \leq_\pi v} \min(c(u,v), c(v,u))$ is independent of π, an optimal solution to OSCM is also an optimal solution to wDFAS on H_G, and vice versa. As an optimal weighted feedback arc set on a graph may be computed separately on each strongly connected component, we get our second pre-processing rule:

▶ **Rule 2.** *Compute solutions for each strongly connected components of H_G independently, and concatenate the results following a topological ordering of the components.*

1.2.0.1 Implementation aspect: computation of SCCs

We compute strongly connected components using the Tarjan algorithm, starting from its rust implementation available at `https://github.com/TheAlgorithms/Rust`. It appears that a crucial optimisation was to never build the graph entirely.

To explain this, let us introduce, given an instance G, π_A of OSCM and $u \in B$, the numbers ℓ_u and r_u, as respectively the smallest and largest positions of a neighbor of u in π_A. For each u, $[\ell_u, r_u]$ is the interval in A where neighbors of u stand. Clearly, for $u, v \in B$ if $r_u \leq \ell_v$, then $c(u,v) = 0$. But unless they have no neighbors (which is never the case in the PACE instances), $c(v,u) > 0$. Therefore, in H_G, there is an edge from u to v.

For pairs of vertices u, v such that $[\ell_u, r_u]$ and $[\ell_v, r_v]$ intersect, we do build an adjacency list dictionary (hash-map in rust). This dictionary does not cover the out-neighbors w of a vertex u such that $r_u \leq \ell_w$. For those, we build a data-structure storing, for each position t in π_A, the sets of vertices x of B such that $\ell_x \geq t$. We use this data-structure to complete iterations over the out-neighbors of a given vertex, whenever needed.

2 Algorithmic description of the solver

An overview of our solver in pseudo-code is given on Algorithm 1. The approach is the same for the exact and parameterized tracks. In the case of the parameterized track, we simply ignore the order that is given.

2.1 Kobayashi-Tamaki

The base of our solver is a Rust implementation of the dynamic programming described in [8], which we denote in the remainder of this description as the "Kobayashi-Tamaki" algorithm. This algorithm is a dynamic procedure over an interval embedding (or path decomposition)

Algorithm 1 Algorithmic overview of the sherby solver, for both the parameterized and the exact track. Θ is a constant hard-coded in the implementation.

```
1  Function sherby-solver($G, \pi_A$)
2      $G$, twins = twin-rule($G$);
3      $SCCs$ = scc-rule($G$);
4      solution = []
5      for $C \in SCCs$ do
6          if exec-time-kobayashi-tamaki($G$) $\leq \Theta$ then
7              solution+=kobayashi-tamaki($C$);
8          else
9              solution+=dfas-SAT($C$);
10     solution = re-insert-twins(solution, twins);
11     return solution;
```

of the instance (see [8] for details). Our implementation of this dynamic programming scheme is iterative (not memoization), computing all entries of the dynamic programming table in an appropriate order. Therefore, given the interval embedding, the run-time of the algorithm is roughly known (function exec-time-kobayashi-tamaki in Algorithm 1). We have therefore hard-coded a threshold Θ on this estimated run-time, and run this approach only if the estimated run-time is below Θ.

2.2 SAT-solver

If the estimated run-time is too large, we resort to a SAT-based approach solving the equivalent wDFAS instance (as explained in the previous Section). This SAT-based solver is a simplified version of [7, 6], the winning submission of PACE 2022, whose target problem was directed feedback vertex set.

We now briefly describe the approach. We first give a SAT formulation for wDFAS. Given a directed graph H, and $\mathcal{C}$ the set of directed cycles of H, we denote the formula $\phi(H)$. It contains:

- one variable x_{uv} for each edge $u \to v$. If set to true, the corresponding edge is included in the feedback arc set.
- one "hard" clause $\bigvee_{u \to v \in C} x_{uv}$ for each cycle $C \in \mathcal{C}$, enforcing that each cycle in the graph must be covered by one edge in the feedback arc set.
- one "soft" clause $\neg x_{uv}$, with a weight of $c(v,u) - c(u,v)$, that expresses the fact that a feedback arc set of minimum weight must be found.

The equivalence with wDFAS is relatively straightforward: if none of the hard clauses are broken, then $\{u \to v \mid x_{uv} = \text{true}\}$ is a feedback arc set. If a set of soft clauses of minimum weight is broken, then this feedback arc set is of minimum weight.

2.2.0.1 The iterative approach of [7]

The winning submission of PACE 2022 [6, 7] used this SAT formulation (for the very similar directed feedback vertex set problem) with an additional trick. It restricts the set of cycles to being of a given maximum length ℓ, and restricts the clauses of the SAT instance to this set. We denote it $\mathcal{C}_\ell$, and the corresponding SAT formula $\phi_\ell(H)$. A crucial property verified by $\phi_\ell(H)$ is the following. Therein, an assignment is said to "break a cycle" if $x_{uv} = \mathsf{true}$ for at least one edge $u \to v$ of that cycle.

▶ **Proposition 2.** *Given H and any integer ℓ, if an optimal assignment for $\phi_\ell(H)$ happens to break all cycles in H, then it is an optimal satisfying assignment for $\phi(H)$.*

Proof. Let OPT and OPT_ℓ be respectively the weights of an optimal assignment to $\phi(H)$ and $\phi_\ell(H)$

Let X be an optimal assignment for $\phi(H)$. As it breaks all cycles in H, it breaks all cycles in $\mathcal{C}_\ell$. It is therefore a valid assignment to $\phi_\ell(H)$, and $\mathsf{OPT}_\ell \leq \mathsf{OPT}$.

If an optimal assignment for $\phi_\ell(H)$ happens to break all cycles in H, we also have $\mathsf{OPT}_\ell \geq \mathsf{OPT}$, and $\mathsf{OPT} = \mathsf{OPT}_\ell$. it is therefore also an optimal assignment for $\phi(H)$. ◀

The overall approach of [6, 7], which is also what we have implemented, is therefore to start with a small value of ℓ, solve $\phi_\ell(H)$ with a state-of-the-art weighted MAX-SAT solver (here, EvalMaxSAT [1]), and check if it covers all cycles in H. If it does, a solution has been found, if it does not, ℓ is increased to $\ell + 1$. A nice property of EvalMaxSAT in this context is its ability to keep some of the work spent on $\phi_\ell(H)$ as a basis for solving $\phi_{\ell+1}(H)$. The pseudo-code of our SAT approach for wDFAS is given on Algorithm 2. It is a simplified version of [6, 7].

Algorithm 2 Pseudo-code of our implementation of a SAT-based approach for wDFAS. It is a simplified version of the winning solver of PACE 2022 [6, 7], which also included many pre-processing rules, and exploited the inner workings of the SAT solver more directly.

```
1 Function dfas-SAT($G, \pi_A$)
2     $H$ = directed-graph($G, \pi_A$);
3     $\ell = 4$                              // start with a small value;
4     $X$ = optimal-assignment($\phi_\ell(H)$)        // using EvalMaxSAT [1];
5     while $X$ does not break all cycles in $H$ do
6         $\ell$+=1;
7         $X$ = optimal-assignment($\phi_\ell(H)$);
8     return topological-order($H \setminus \{u \to v \mid x_{uv} = \mathsf{true}\}$);
```

3 Results and Conclusion

Table 1 summarizes the numbers of instances solved by our solvers within the allocated constraints (8GB of RAM and 30 minutes of CPU time per instance). Perhaps thanks to the application of Rules 1 and 2, our Rust implementation of the Kobayashi-Tamaki algorithm [8] solves almost all of the 125 public instances of the parameterized track. It only needs to resort to the SAT solver for two of them, 122 and 123. All 100 instances in the set that `https://optil.io` used are solved almost instantly, none of them requiring the SAT-solver. Table 2 also includes the run-times as reported by optil.

References

1 Florent Avellaneda. A short description of the solver evalmaxsat. *MaxSAT Evaluation*, 8, 2020.

2 Alexander Dobler. A note on the complexity of one-sided crossing minimization of trees. *arXiv preprint arXiv:2306.15339*, 2023.

solver	exact track	parameterized track
kobayashi-tamaki [8] alone	51/100	122/125
Algorithm 1 (kobayashi-tamaki+dfas-SAT)	67/100	125/125

Table 1 Number of instances in the public sets solved by our Rust implementation of the Kobayashi-Tamaki algorithm and our overall solver (Algorithm 1). These numbers include the applications of Rules 1 and 2. In the parameterized track, the two instances on which Kobayashi-Tamaki failed were 122 and 123, (but they are solved by the SAT solver in about a minute).

track	Successes	RTE	MLE	total cumulated time (on successes)
exact	67	32	1	56709 seconds
parameterized	100	0	0	3.27 seconds

Table 2 Results on the public data sets as per the `https://optil.io` website. RTE stands for "Run-Time limit Exceeded" (30 minutes per instance) and MLE for "Memory Limit Exceeeded" (8GB per instance). Run-times may be slightly different from the final evaluation. Note that for the parameterized track, only a 100 instances were included on optil, excluding the 122 and 123, the only which required the SAT solver. These results, for the parameterized track, are therefore solely from our Rust implementation of Kobayashi-Tamaki [8].

3 Vida Dujmović, Henning Fernau, and Michael Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. *Journal of Discrete Algorithms*, 6(2):313–323, 2008.

4 Vida Dujmovic and Sue Whitesides. An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. *Algorithmica*, 40:15–31, 2004.

5 Henning Fernau, Fedor V Fomin, Daniel Lokshtanov, Matthias Mnich, Geevarghese Philip, and Saket Saurabh. Ranking and drawing in subexponential time. In *International Workshop on Combinatorial Algorithms*, pages 337–348. Springer, 2010.

6 Rafael Kiesel and Andre Schidler. DAGger - An Exact Directed Feedback Vertex Set Solver, June 2022. `doi:10.5281/zenodo.6627405`.

7 Rafael Kiesel and André Schidler. A dynamic maxsat-based approach to directed feedback vertex sets. In *2023 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 39–52. SIAM, 2023.

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.