

# The PACE 2024 Parameterized Algorithms and Computational Experiments Challenge: One-Sided Crossing Minimization

Philipp Kindermann ✉ 

Trier University, Germany

Fabian Klute ✉ 

Polytechnic University of Catalonia, Barcelona, Spain

Soeren Terziadis ✉ 

Eindhoven University of Technology, The Netherlands

---

## Abstract

This article is a report by the challenge organizers on the 9th Parameterized Algorithms and Computational Experiments Challenge (PACE 2024). As was common in previous iterations of the competition, this year's iteration implemented an exact and heuristic track for a parameterized problem that has gained attention in the theory community. This year's challenge is about the ONE-SIDED CROSSING MINIMIZATION PROBLEM (OSCM). In the exact track, the competition participants were asked to develop an exact algorithm that can solve as many instances as possible from a benchmark set of 100 instances – with a time limit of 30 minutes per instance. In the heuristic track, the task must be accomplished within 5 minutes, however, the result in this track is not required to be optimal. New this year is the parameterized track, which has the same rules as the exact track, but instances are guaranteed to have small cutwidth. As in previous iterations, the organizers handed out awards to the best solutions in all tracks and to the best student submissions.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph algorithms

**Keywords and phrases** One-Sided Crossing Minimization, Algorithm Engineering, FPT, Heuristics

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2024.26

**Acknowledgements** The prize money (€4000) was generously provided by Networks [33], an NWO Gravitation project of the University of Amsterdam, Eindhoven University of Technology, Leiden University and the Center for Mathematics and Computer Science (CWI). We are grateful to the whole optil.io team, especially to Jan Badura for the fruitful collaboration and for hosting the competition at the optil.io online judge system. We also thank Markus Wallinger, who made his exact solver available to the organizers prior to the competition for internal evaluations [38].

## 1 Introduction: History and Timeline of PACE

The Parameterized Algorithms and Computational Experiments Challenge (PACE) was conceived in Fall 2015 to deepen the relationship between parameterized algorithms and practice. The declared mission of the PACE challenge is to

- bridge the divide between the theory of algorithm design and analysis, and the practice of algorithm engineering,
- inspire new theoretical developments,
- investigate in how far theoretical algorithms from parameterized complexity and related fields are competitive in practice,
- produce universally accessible libraries of implementations and repositories of benchmark instances,
- encourage the dissemination of these findings in scientific papers.



© Philipp Kindermann, Fabian Klute, and Soeren Terziadis;  
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. 26; pp. 26:1–26:20

Leibniz International Proceedings in Informatics



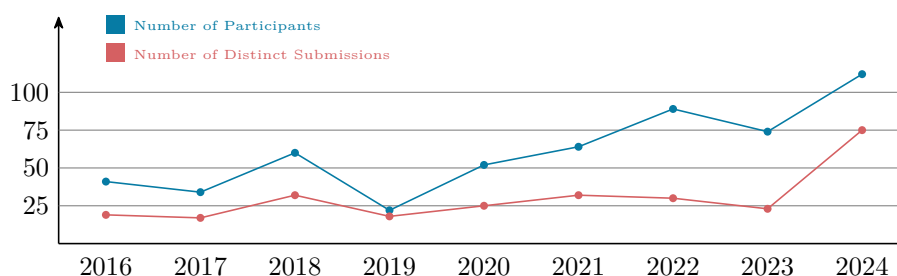
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 26:2 PACE 2024: One-Sided Crossing Minimization

Previous iterations of the PACE challenge addressed a variety of graph optimization problems. Specifically, the previous iterations considered

<b>PACE 2016</b>	TREEWIDTH and UNDIRECTED-FEEDBACK-VERTEX-SET	[6];
<b>PACE 2017</b>	TREEWIDTH and MINIMUM FILL-IN	[7];
<b>PACE 2018</b>	STEINER TREE	[5];
<b>PACE 2019</b>	VERTEX-COVER and HYPERTREEWIDTH	[13];
<b>PACE 2020</b>	TREEDEPTH	[28];
<b>PACE 2021</b>	CLUSTER-EDITING	[25];
<b>PACE 2022</b>	DIRECTED-FEEDBACK-VERTEX-SET	[18];
<b>PACE 2023</b>	TWINWIDTH	[3].

Several of the previous iterations also contained more specialized tracks. Starting with the first iteration of PACE, many participants from all over the world were interested in the challenge and quickly established PACE as a highly competitive challenge. The competition attracted a record number of 112 participants this year, resulting in 75 submissions; see Figure 1.

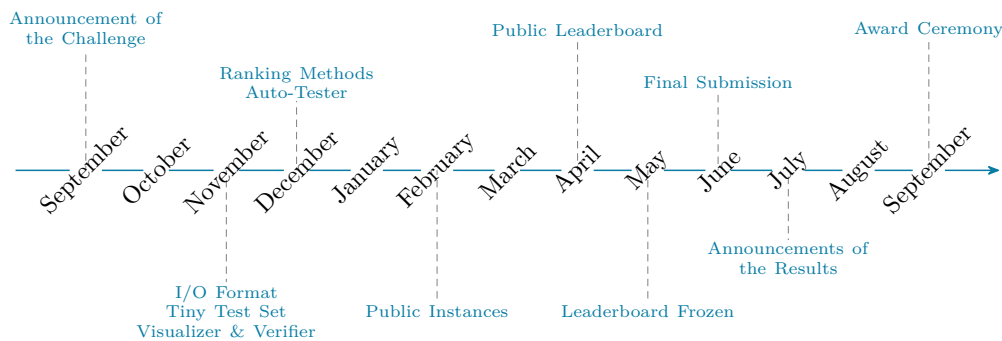


■ **Figure 1** Overview of the number of participants and distinct submissions of the PACE challenge over the years. Teams submitting multiple times and to multiple tracks are counted multiple times.

Papers inspired by concrete implementations created in the context of the PACE challenge were published in prestigious conferences such as ACDA, ALENEX, ESA (Track B), SEA, and WADS. The instances provided by PACE have also often been used to showcase further algorithmic improvements by being used as an established benchmark, ranging also to other competitions such as the famous SAT competition [2].

This report contains the relevant information on the ninth PACE challenge. The problem chosen was ONE-SIDED CROSSING MINIMIZATION, an important classical graph drawing problem with applications in hierarchical graph drawing, e.g., in the Sugiyama framework [37]. The challenge featured three tracks: an exact track, a heuristic track and a parameterized track. In the exact track, the task was to find an optimal solution of a given instance within 30 minutes and a memory limit of 8 GB. The instances in the exact track were guaranteed to have a low number of crossings relative to the instance size. In the heuristic track the number of crossings could be very large and the task was to compute a valid (but not necessarily optimal) solution with as few crossings as possible within a time limit of 5 minutes and a memory limit of 8 GB. The parameterized track used the same rules as the exact track (compute an optimal solution within 30 minutes using 8GB). However here the number of crossings was not guaranteed to be small and instead the cutwidth of the given instances was low. Additionally a witness for this low cutwidth was provided for every instance.

The timeline of the challenge started with the announcement in September 2023. Details about the input and output format were provided in November 2023 together with a tiny test set to allow the participants to start with the challenge. In addition, a small visualizer was provided, which can be used to view instances and solutions. Alongside we provided (via pip and source download) a verifier, which could be used to verify a given solution using a set of crossing counting algorithms. The concrete ranking methods for both tracks were published in December 2023. At the same time we provided a JUnit-like auto-tester, which runs a solver against a given set of instances and compares the output to provided (optimal) solutions. In early February 2024 the public instances and details about the benchmark set were published. Additionally a Github repository was made available with the intention that participants could add their own created test instances to make them available to all participating teams. The public leaderboard on the optil.io platform was opened in April 2024 and frozen on May 20th 2024. This allowed the participants to test their solvers on the public instances and provided a provisional ranking. The final version of the submission for the solver code was due on the ninth of June 2024 and the descriptions of the solvers had to be submitted until June 23. Afterwards, the submissions were evaluated on the private instances, which were similar in structure to the public instances but unknown to the participants. The results of this evaluation were announced in July 2024 (a correction of the ranking was issued on the ninth of August 2024), and the award ceremony took place during the International Symposium on Parameterized and Exact Computation (IPEC) 2024 at Royal Holloway University of London in Egham. The complete timeline can be found in Figure 2.



■ **Figure 2** Timeline of the PACE challenge in 2024 (the diagram ranges from September 2023 to September 2024). The next iteration of the PACE challenge for 2025 was announced during the award ceremony.

## 2 The Challenge Problem: One-Sided Crossing Minimization

This year's challenge was about the problem ONE-SIDED CROSSING MINIMIZATION (OSCM). This problem involves arranging the nodes of a bipartite graph on two layers (typically horizontal), with one of the layers fixed, aiming to minimize the number of edge crossings. More formally:

## ONE-SIDED CROSSING MINIMIZATION (OSCM)

**Input:** A bipartite graph  $G = ((A \dot{\cup} B), E)$ , and a linear order of  $A$ .

**Output:** A linear order of  $B$ .

**Measure:** The number of edge crossings in a straight-line drawing of  $G$  with  $A$  and  $B$  on two parallel lines, following their linear order.

OSCM is one of the basic building blocks used for drawing hierarchical graphs [37]. While easy to state it turns out that the problem is difficult to solve efficiently. As many graph drawing problems OSCM is well known to be NP-hard [14], even for star forests of degree 4 [32] and for trees [9]. Obtaining exact solutions in practice is commonly done using SAT or ILP formulations of the problem [23].

Turning to other exact algorithms, the problem can be solved in FPT time using the number of crossings as the parameter. Dujmovic and Whitesides [11] gave the first algorithm in 2004. Subsequent research [12, 26] pushed the running time down to  $O(k2^{\sqrt{2k}} + n)$ , where the exponent  $\sqrt{2k}$  is asymptotically optimal assuming the exponential time hypothesis.

Aside from exact solutions, OSCM does admit a constant-factor approximation [15]. More importantly for practical considerations though, OSCM admits good heuristics. Two of the best-known ones are the *barycenter* and the *median heuristic*. As the names suggest, in the former each vertex is placed in the barycenter of its neighbors and in the latter in the median position of its neighbors. These very simple heuristics even come with some theoretical guarantees. For example, the barycenter heuristic yields a  $O(\sqrt{n})$  approximation [31].

For an extended overview, see Chapter 13.5 of the Handbook of Graph Drawing [22].

In the parameterized track, all instances have small cutwidth. Given a graph  $G = (V, E)$  and a linear ordering  $\pi$  on  $V$ , the *cutwidth* of  $(G, \pi)$  is the maximum number of edges that cross any partitioning of  $V$  into earlier and later subsets of  $\pi$ , that is,  $\max_{1 \leq i < |V|} |\{(u, v) \in E \mid \pi(u) \leq i < \pi(v)\}|$ . The *cutwidth* of  $G$  is the minimum cutwidth over all possible linear orderings of  $V$ .

The cutwidth of a graph can be computed in FPT time using the cutwidth as the parameter. In particular, there is an  $2^{O(k^2)}n$ -time algorithm to compute the cutwidth  $k$  if a graph [17]. There is a connection between cutwidth and the general crossing number of graphs: Any graph  $G = (V, E)$  with cutwidth  $k$  requires at least  $\frac{k^2}{1176} - \sum_{v \in V} \left(\frac{\deg(v)}{4}\right)^2$  crossings in any drawing [8].

Since OSCM is NP-hard even for star forests of degree 4 [32], which have cutwidth 2, it is paraNP-hard if parameterized by the cutwidth. However, in the input to OSCM, the order of  $A$  is fixed, and the instances of the NP-hardness proof come with orders of  $A$  that do not admit a constant cutwidth, that is, there is no linear order on the vertices of the constructed graphs where the order of  $A$  is the same as in the input and the cutwidth is constant. Hence, we are interested in studying OSCM for graphs where the cutwidth remains small even if the order of  $A$  has to remain the same as in the input.

### 3 The Setup of PACE 2024

As already mentioned before, this year's challenge featured three tracks. The *exact* and *parameterized* track both required the computation of an optimal solution and gave different guarantees about the structure of the input instances. In the *heuristic* track, no guarantees were made about the structure of the instances and participants were tasked with computing the best (but not necessarily optimal) layout they can find within a more limited time frame.

### 3.1 The Exact Track

The task of this track was to compute an optimal solution of OSCM for 200 graphs, 100 of which were public and 100 of which were not known by the participants and were only presented to the solver during the evaluation in a compartmentalized judge system. For each of the graphs, the solver had a time limit of *30 minutes* and a memory limit of *8 GB* to output a solution. All instances were guaranteed to have a “not too large” number of crossings in an optimal solution (detailed information about this can be found in Section 3.5).

The organizers of the competition encouraged submissions that implement provably optimal algorithms, however, this was not a formal requirement. The exact rule stated on the website was

*Submissions should be based on provably optimal algorithms, however, this is not a formal requirement. Submissions that output an incorrect solution or a solution that is known to be non-optimal will be disqualified. Besides dedicated algorithms, we also encourage submissions based on other paradigms such as SAT, MaxSAT, or ILPs.*

The requirement of outputting optimal solutions extends to instances that were not included in the set of the 200 evaluation instances. In the exact track 7 solvers were disqualified, 5 on the basis of outputting wrong answers for one or more of the 200 evaluation instances and 2 due to subsequently found counterexample instances.

Submissions of this track were ranked by the *number of solved instances*. In case of a tie, the winner was determined by the *total time spent on the solved instances*. In particular, there was no need to abort a “hopeless” run early.

### 3.2 The Heuristic Track

In this track, the solvers were tasked with computing a good solution quickly. The solvers were run on each instance for *5 minutes* and received the Unix signal SIGTERM afterwards. When receiving this signal, the solver had to output a valid layout in the defined solution format immediately to the standard output and terminate. If the program did not halt in a reasonable time after receiving the signal, it was stopped via SIGKILL and the instance was counted as time limited exceeded. The memory limit for this track was *8 GB* as well. For this track, solutions did not have to be optimal.

Submissions were ranked by the sum over all instances of

$$\frac{\# \text{ crossings in solver layout}}{\text{smallest } \# \text{ crossings known to the PC}}$$

Note that the “smallest number of crossings in any layout known to the PC” may not be optimal, i. e., may be larger than the number of crossings in an optimal solution.

### 3.3 The Parameterized Track

This track had the same rules as the Exact Track. However, the instances here could require a large number of crossings, but they had small cutwidth: there is an ordering of the vertices of the graph such that every cut obtained by partitioning the vertices into earlier and later subsets of the ordering is crossed by at most “a small number” of edges. Such an ordering was provided in the input. Note that in this ordering the vertices of *A* and *B* were generally interleaved, but the order of the vertices of *A* (the fixed side) was the same as in the problem instance.

### 3.4 Internal Solver

One of the most challenging aspects of creating the benchmark set was to strike a balance between sufficiently difficult instances and the need to solve them to optimality in order to judge the exact and parameterized track. The organizers used an ILP solver to answer this question. This solver was implemented based on the formulation of Jünger and Mutzel [23].

Specifically, let  $A$  and  $B$  be the two partite sets of the given bipartite graph. For every pair of vertices  $v_i, v_j \in B$  we create a binary variable  $t_{i,j}$ , which should be true if and only if  $v_i$  appears before  $v_j$  in the final order of  $B$ . Since this order is linear it is of course transitive. This can be enforced by adding for every  $i < j < k$  the inequalities  $0 \leq t_{i,j} + t_{j,k} - t_{i,k} \leq 1$ , i.e., transitivity constraints. The number of crossings between the edges incident to  $v_i$  and the edges incident to  $v_j$  is only dependent on the relative order of  $v_i$  and  $v_j$  and can easily be precomputed. Let  $c_{i,j}$  ( $c_{j,i}$ ) be this number if  $v_i$  appears before (after)  $v_j$  in the final order of  $B$ . Then we simply minimize  $\sum_{i < j} (c_{i,j}t_{i,j} + c_{j,i}(1 - t_{i,j}))$ .

Since the number of transitivity constraints can grow quite large, the internal solver of the organizers added them on demand using callbacks whenever a solver found a new solution. The solver was initialized without any transitivity constraints. If a new solution is found, the binary variables are used to define a directed graph on all vertices of  $B$ , i.e., an edge is added from  $v_i$  to  $v_j$  if  $t_{i,j}$  is true otherwise the edge in the opposite direction is added. If this graph contains any cycles, we obtain all chordless cycles (which are necessarily triangles) and add the transitivity constraint for the three involved vertices.

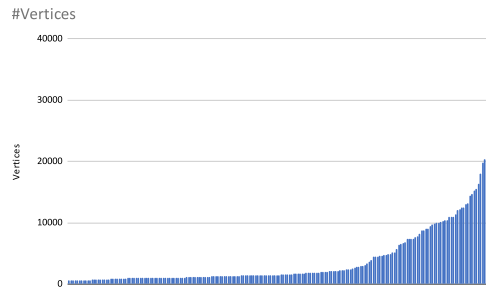
### 3.5 Benchmark Set

The fourth aim of the PACE challenge was to *produce universally accessible libraries of implementations and repositories of benchmark instances*. While the first part of this aim was exactly what we expected from the participants, it was the duty of the program committee to produce the benchmark instances. The properties of the benchmark instances we strived were that

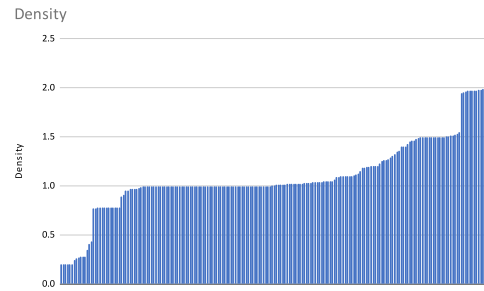
1. the benchmark instances should be heterogeneous,
2. they vary in size and difficulty and
3. it remains a challenging benchmark after the challenge.

The reason for the first criterion was that we wanted to evaluate the overall performance of the approaches developed by the participants (and not the performance on, say, a specific graph class). The goal of the second criterion was to make the challenge interesting and fun. We wanted a benchmark set in which every participant can solve at least a few instances, which should especially encourage student teams to participate as well. The medium instances were the ones that were meant to distinguish the quality of the various solvers, and the hard instances ensured that the tracks which require optimal solutions could be judged based on the number of solved instances. Moreover, we wanted to create a test set which contained instances hard enough to remain interesting after the challenge, instead of one that is simply “solved” after the competition. We expected that these hard instances are barely solvable by solvers developed in the time span of the competition and, thus, leave room for further research.

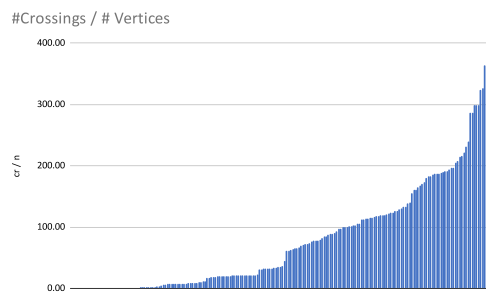
We created generators for several graph classes, including uniform random (planar) graphs, cycles, paths, complete bipartite graphs, stars, matchings, trees, lobsters, (double-)caterpillars, grids, quadrangulations, (partial)  $k$ -trees, wheels, disk intersection graphs, interval bigraphs, (circular) ladders, hypercubes, co-graphs, intersection graphs, bipartite permutation graphs, and graphs with small vertex cover / cutwidth / neighborhood diversity.



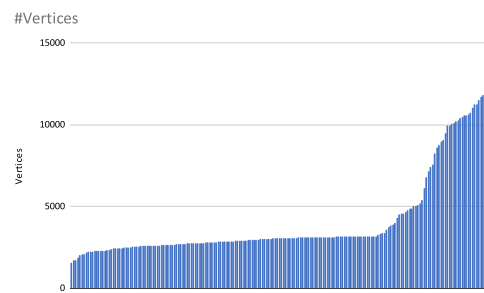
(a) Distribution of #vertices in the exact benchmark set.



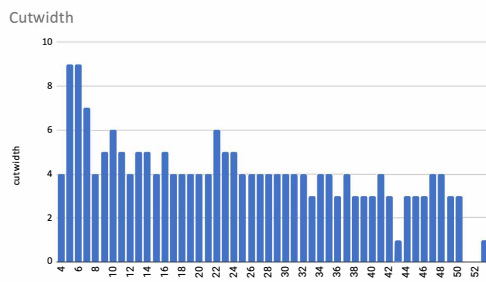
(b) Distribution of edge density in the exact benchmark set.



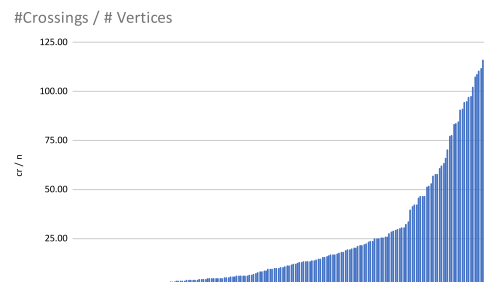
(c) Distribution of crossing density in the exact benchmark set.



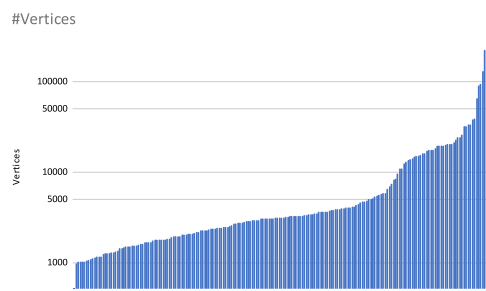
(d) Distribution of #vertices in the parameterized benchmark set.



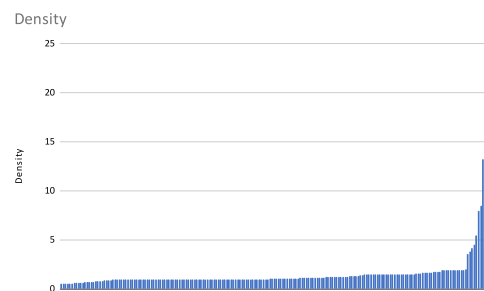
(e) Number of instances by cutwidth in the parameterized benchmark set.



(f) Distribution of crossing density in the parameterized benchmark set.



(g) Distribution of #vertices in the heuristic benchmark set (log scale).



(h) Distribution of edge density in the heuristic benchmark set.

■ **Figure 3** Details about the benchmark sets in the three tracks.



We also included (variations of) known examples from the literature that common heuristics perform badly on. To further increase variety, we also added options to randomly permute one or both bipartitions and to glue instances together. Many generators are based on Networkx [21]. The code can be found online<sup>1</sup>.

To find a good balance in the difficulty of the benchmark, we created about 15,000 instances of various sizes from each of these graph classes and tried to solve them with our solver. For the exact and parameterized track, we only selected instances for which we knew the optimum solution.

For the exact track, we decided to use between 5 and 15 instances from each graph class. We used 60 instances that our solver could solve within 10 minutes, 50 instances that it could solve in under 20 minutes, 40 instances that it could solve in under 30 minutes, and 50 instances that required more time. While general instances can require  $\Omega(m^2) \in \Omega(n^4)$  crossings, we selected instances where the *crossing density* (i.e., the number of crossings required divided by the number of vertices) does not get too large; see Figure 3c. The graphs had between 560 and 20,000 vertices (with one exception that has 32,691 vertices) and edge density between 0.2 and 2.5; see Figures 3a and 3b.

For the heuristic track, we used similar instances, but made them much larger; see Figures 3g and 3h. The graphs had up to 262,124 vertices and 1,114,112 edges.

For the parameterized track, we also used only instances for which we knew the optimum solution. We generated instances with bounded cutwidth by two random sampling approaches. All but two instances had cutwidth at most 50; see Figure 3e. The graphs had between 1552 and 13,674 vertices and required between  $0.08n$  and  $123.78n$  crossings.

### 3.6 Available Tools

#### Mini Test Set

As a first set of instances and to get development for the participants off the ground, we provided a set of mini test instances<sup>2</sup>. We also provided the solutions to these test instances<sup>3</sup>. An impression of these instances can be seen in Figure 4.

#### Verifier

We provided a Python-based tool to verify a solution and test a given solver against a set of tests. This verifier was provided as a python package on the website<sup>4</sup> and was continuously updated throughout the challenge. To verify a produced solution two main algorithms were available. The first one was a simple, inefficient, but surely correct implementation that just tested for every pair of edges whether they cross. The second method and usually the default option, was an algorithm based on so-called segment trees. For comfort the verifier could not only be run on a single solution to verify it, but instead could be provided with a folder of test instances and a solver. Each instance was then solved using the provided solver and the resulting solution checked. By default the test instances used in this verifier mode were the tiny test set described above.

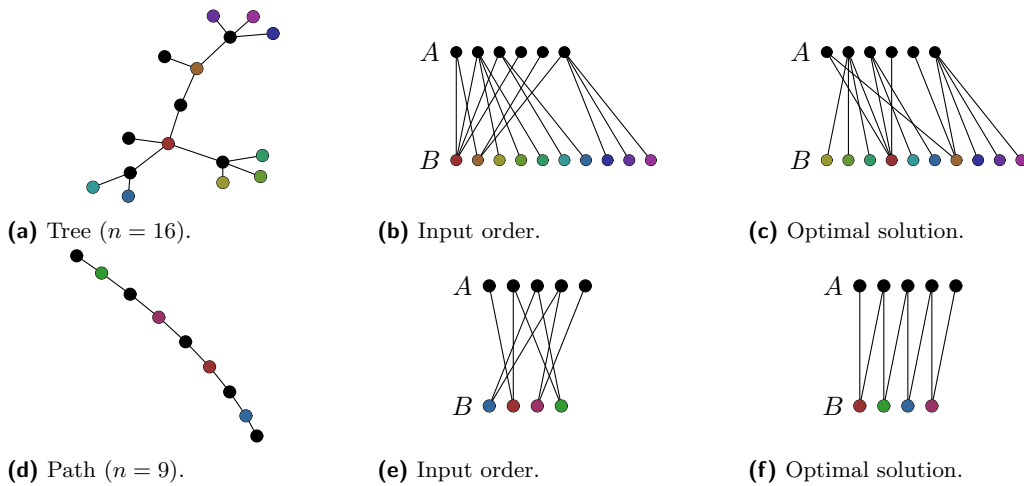
<sup>1</sup> [https://pacechallenge.org/2024/instance\\_generator.py](https://pacechallenge.org/2024/instance_generator.py)

<sup>2</sup> [https://pacechallenge.org/2024/tiny\\_test\\_set.zip](https://pacechallenge.org/2024/tiny_test_set.zip)

<sup>3</sup> [https://pacechallenge.org/2024/tiny\\_test\\_set-sol.zip](https://pacechallenge.org/2024/tiny_test_set-sol.zip)

<sup>4</sup> <https://pacechallenge.org/2024/verifier/>





**Figure 4** Two example instances from the mini test set given with a force directed layout to illustrate the structure as well as the input order and the optimal 2-layered layout. The partite set, whose order is fixed ( $A$ ) is drawn in black, the vertices of the permutable set ( $B$ ) are drawn in unique colors.

## Visualizer

We also provided a visualizer tool to the participants.<sup>5</sup> This tool displayed a given graph and solution in a graphical interface. For small and medium size instances this allowed the participants to get a graphical idea of their solutions and was hopefully helpful in finding potential improvements of their algorithms. The images in Figure 4 were created using this visualizer.

## 4 Participants and Results

There were 25, 32 and 17 distinct submissions to the exact, heuristic and parameterized track, respectively. Hence, in total there were 40 distinct teams with a total number of 112 participants representing four continents and 21 countries, which made this the largest PACE challenge yet. The results are listed below.

### 4.1 Ranking of the Exact Track


The ranking for the exact track is listed subsequently; We list the number of solved instances from the 100 private instances plus the 100 public instances as well as the total computation time used for the solved instances. Submissions marked with an “🎓” icon are student submissions after the following rules

*A student is someone who is not and has not been enrolled in a PhD program before the submission deadline. A submission is eligible for a Student Submission Award if either all its authors are students, or besides student co-author(s) there is one non-student co-author that confirms, at the moment of submission, that a clear majority of conceptual and implementation work was done by the student co-author(s).*

<sup>5</sup> <https://pacechallenge.org/2024/visualizer/>

## 26:10 PACE 2024: One-Sided Crossing Minimization

The first five valid student submissions were eligible for a Student Submission Award; there were three such submissions, which received this award.

- Rank 1** [mppeg](#) solved [199](#) instances in [5682.93](#) seconds. [Link](#) 
- Authors** Michael Jünger, Paul Jünger, Petra Mutzel and Gerhard Reinelt  
**Affiliation** University of Cologne, University of Bonn and Heidelberg University
- Rank 2** [uzl](#) solved [195](#) instances in [7692.89](#) seconds. [Link](#) 
- Authors** Max Bannach, Florian Chudigiewitsch, Kim-Manuel Klein and Marcel Wienöbst  
**Affiliation** European Space Agency and Institute for Theoretical Computer Science, University of Lübeck
- Rank 3** [CRGone](#) solved [192](#) instances in [15520.39](#) seconds. [Link](#) 
- Authors** Alexander Dobler  
**Affiliation** Technische Universität Wien
- Rank 4** [Guilucand](#) solved [187](#) instances in [9358.96](#) seconds. [Link](#) 
- Authors** Andrea Cracco  
**Affiliation** Università degli Studi di Verona
- Rank 5** [crossy](#)  solved [180](#) instances in [19099.31](#) seconds. [Link](#) 
- Authors** Tobias Röhr and Kirill Simonov  
**Affiliation** Hasso Plattner Institute, University of Potsdam
- Rank 6** [weberknecht](#) solved [164](#) instances in [21408.17](#) seconds. [Link](#) 
- Authors** Johannes Rauch  
**Affiliation** Institute of Optimization and Operations Research, Ulm University
- Rank 7** [LUNCH](#) solved [157](#) instances in [10425.52](#) seconds. [Link](#) 
- Authors** Kenneth Langedal, Matthias Bentert, Thorgal Blanco and Pål Grønås Drange  
**Affiliation** University of Bergen
- Rank 8** [Arcee](#)  solved [152](#) instances in [11189.13](#) seconds. [Link](#) 
- Authors** Kimon Boehmer, Lukas Lee George, Fanny Hauser and Jesse Palarus  
**Affiliation** Université Paris-Saclay, Technical University Berlin
- Rank 9** [lcs](#) solved [136](#) instances in [1186.94](#) seconds. [Link](#) 
- Authors** Mohamed Mahmoud Abdelwahab, Faisal N. Abu-Khzam and Lucas Isenmann  
**Affiliation** Lebanese American University
- Rank 10** [sherby](#) solved [135](#) instances in [1683.19](#) seconds. [Link](#) 
- Authors** Manuel Lafond, Alitzel López Sánchez and Bertrand Marchand  
**Affiliation** Department of Computer Science, University of Sherbrooke
- Rank 11** [U\\_OCM](#) solved [121](#) instances in [7907.2](#) seconds. [Link](#) 
- Authors** Mert Biyikli, Kathrin Hanauer, Sophia Heck, Lukas Krumpeck, Lara Ost, Tobias Prisching, Ole Schlüter, Matej Vedak and Maximilian Vötsch  
**Affiliation** Faculty of Computer Science, University of Vienna and Faculty of Physics, University of Vienna
- Rank 12** [roundabout](#) solved [109](#) instances in [564.63](#) seconds. [Link](#) 
- Authors** Emmanuel Arrighi and Petra Wolf  
**Affiliation** EnsL, Univ Lyon, UCBL, CNRS, Inria in Lyon and LaBRI, CNRS, Université de Bordeaux
- Rank 13** [HWoydt](#) solved [75](#) instances in [12.75](#) seconds. [Link](#) 
- Authors** Henning Martin Woydt  
**Affiliation** Heidelberg University

**Rank 14** [GOAT](#) solved 74 instances in 725.36 seconds. [Link](#) 

**Authors** Patrik Drbal, Michal Dvořák, Dušan Knop, Jozef Koleda, Jan Pokorný and Ondřej Suchý

**Affiliation** Czech Technical University in Prague

**Rank 15** [trex-ufmg-ilp](#) solved 41 instances in 3918.34 seconds. [Link](#) 

**Authors** Luis Henrique Gomes Higino, Kaio Henrique Masse Vieira, Alan Prado, Guilherme de Castro Mendes Gomes, Laila Melo Vaz Lopes, Gabriel Ubiratan Barreto Pereira de Oliveira, Gabriel Lucas Costa Martins, Heitor Gonçalves Leite, Matheus Torres Prates, Gabriel Vieira and Vinicius Fernandes dos Santos

**Affiliation** Departamento de Ciência da Computação, Universidade Federal de Minas Gerais

**Rank 16** [studentgroupfuberlin](#)  solved 28 instances in 12844.18 seconds. [Link](#) 

**Authors** Garvin Konopka, Colin Alexander Voigt and Joshua Alexander Hanheiser

**Affiliation** Freie Universität Berlin

The details of the remaining 9 teams, which were either disqualified or did not solve any instance correctly, can be found on the PACE website. Submission [mjdv](#) solved 157 instances in 5000.98 seconds, but was disqualified due to suboptimal output on one instance due to a bug in their code.

## 4.2 Methods used by the Winners of the Exact Track

The **Exact Track winner mpeg** [24] reformulate the problem of the challenge into a linear ordering problem, which they solve using a branch & cut approach [19, 20]. Their method formulates the problem as an integer linear program, which uses binary variables to indicate the order of any ordered pair of vertices. Transitivity constraints enforce that there are no cycles in the ordering implied by a valid solution to the program. In the original formulation, preventing cycles of length three is sufficient, however the approach of mpeg utilizes multiple methods to speed up their approach, by reducing the size of the ILP. As a result additional cycles have to be prevented via constraint. The speed-up methods include among others the decomposition of instances into connected components and fixing any relative order between vertices. For instances where it is feasible, they additionally compute an initial solution using the “Kernigham-Lin 2” heuristic [35].





















The **runner-up uzl** [4] observed that the ILP formulation is equivalent to the WEIGHTED FEEDBACK ARC SET problem, which can also be expressed as a HITTING SET problem, where all cycles are sets. This team’s approach is based on existing formulations for WEIGHTED FEEDBACK ARC SET [1, 19]. Initially only some cycle constraints are added. After finding a solution to a relaxation of their model, a heuristic is used to identify new cycle constraints until all cycles are removed.

The **third-place CRGone** [10] again uses an ILP formulation after employing some reduction rules to decrease the instance size. This is, e.g., done by contracting vertices with the same neighborhood or (similar to the winning team) fixing ordering variables, if one of the two relative orders creates no crossings. The model is initialized without any transitivity constraints between binary variables, which are separated in a predefined order, preferring constraints for vertices whose neighbors do not interleave.








### 4.3 Ranking of the Heuristic Track

The ranking for the heuristic track is listed subsequently; We list the score for the 100 private instances plus the 100 public instances, computed as described above, as well as the total computation time. The larger the score, the better. Submissions marked with an “🎓” icon are student submissions (using the same rules as above) of which the top five obtained a Student Submission Award.

- Rank 1** [CIMAT\\_Team](#) got a score of [199.99996](#) in [59236.67](#) seconds. [Link](#) 🎓  
**Authors** Carlos Segura, Lázaro Lugo, Gara Miranda and Edison David Serrano Cárdenas  
**Affiliation** Area de Computación, Centro de Investigación en Matemáticas (CIMAT) in Mexico, Departamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna and Area de Matemáticas Aplicadas, Centro de Investigación en Matemáticas (CIMAT) in Mexico
- Rank 2** [LUNCH](#) got a score of [199.99994](#) in [80545.43](#) seconds. [Link](#) 🎓  
**Authors** Kenneth Langedal, Matthias Bentert, Thorgal Blanco and Pål Grønås Drange  
**Affiliation** University of Bergen
- Rank 3** [Martin\\_J\\_Geiger](#) got a score of [199.99983](#) in [41662.87](#) seconds. [Link](#) 🎓  
**Authors** Martin Josef Geiger  
**Affiliation** University of the Federal Armed Forces Hamburg
- Rank 4** [Arcee](#) 🎓 got a score of [199.99998](#) in [38339.44](#) seconds. [Link](#) 🎓  
**Authors** Kimon Boehmer, Lukas Lee George, Fanny Hauser and Jesse Palarus  
**Affiliation** Université Paris-Saclay, Technical University Berlin
- Rank 5** [guilhermefonseca](#) got a score of [199.99978](#) in [26336.25](#) seconds. [Link](#) 🎓  
**Authors** Guilherme D. da Fonseca  
**Affiliation** LIS, Aix-Marseille Université
- Rank 6** [Bob](#) got a score of [199.99978](#) in [27380.75](#) seconds. [Link](#) 🎓  
**Authors** Sergey Pupyrev  
**Affiliation** Menlo Park
- Rank 7** [uzl](#) got a score of [199.9996](#) in [80644.98](#) seconds. [Link](#) 🎓  
**Authors** Max Bannach, Florian Chudigiewitsch, Kim-Manuel Klein and Marcel Wienöbst  
**Affiliation** European Space Agency and Institute for Theoretical Computer Science, University of Lübeck
- Rank 8** [slimmer](#) got a score of [199.99865](#) in [54761.13](#) seconds. [Link](#) 🎓  
**Authors** Steffen Limmer and Nils Einecke  
**Affiliation** Honda Research Institute Europe GmbH
- Rank 9** [UAIC\\_OCM](#) 🎓 got a score of [199.99735](#) in [56047.38](#) seconds. [Link](#) 🎓  
**Authors** Andrei Arhire, Eugen Croitoru, Matei Chiriac and Alex Dumitrescu  
**Affiliation** Alexandru Ioan Cuza University of Iași
- Rank 10** [axs](#) 🎓 got a score of [199.99037](#) in [59409.74](#) seconds. [Link](#) 🎓  
**Authors** Chenghao Zhu, Yi Zhou and Bo Peng  
**Affiliation** University of Electronic Science and Technology of China and Southwestern University of Finance and Economics Chengdu
- Rank 11** [weberknecht](#) got a score of [199.97621](#) in [6187.64](#) seconds. [Link](#) 🎓  
**Authors** Johannes Rauch  
**Affiliation** Institute of Optimization and Operations Research, Ulm University

- Rank 12 [tlopez](#)  got a score of [199.93344](#) in [80556.06](#) seconds. [Link](#) 
- Authors** Toan Lopez and Florian Sikora  
**Affiliation** Student of Université Paris Dauphine
- Rank 13 [KongQi](#)  got a score of [199.66556](#) in [38664.61](#) seconds. [Link](#) 
- Authors** Qi Kong, Zhouxing Su and Zhipeng Lü  
**Affiliation** Huazhong University of Science and Technology
- Rank 14 [heiCross](#)  got a score of [199.46119](#) in [80480.59](#) seconds. [Link](#) 
- Authors** Adil Chhabra, Marlon Dittes, Alvaro Garmendia, Ernestine Großmann, Tomer Haham, Shai Peretz, Henrik Reinstädler, Antonie Wagner and Henning Woydt  
**Affiliation** University of Heidelberg
- Rank 15 [LOPP](#)  got a score of [198.93312](#) in [80720.08](#) seconds. [Link](#) 
- Authors** Arijeet Pramanik, Rishabh Dev, Vimal Narassimmane and Srinibas Swain  
**Affiliation** Department of Computer Science and Engineering, IIIT Guwahati
- Rank 16 [GAON](#)  got a score of [198.88211](#) in [80720.77](#) seconds. [Link](#) 
- Authors** Rishabh Dev, Arijeet Pramanik, Vimal Narassimmane and Srinibas Swain  
**Affiliation** Department of Computer Science and Engineering, IIIT Guwahati
- Rank 17 [ericweidner](#)  got a score of [198.79449](#) in [4961.61](#) seconds. [Link](#) 
- Authors** Carolin Rehs and Eric Weidner  
**Affiliation** Technical University of Dortmund
- Rank 18 [KUL-TW](#) got a score of [198.56896](#) in [19684.34](#) seconds. [Link](#) 
- Authors** Tony Wauters and Fabien Nießen  
**Affiliation** NUMA, Department of Computer Science, KU Leuven
- Rank 19 [DRIP](#)  got a score of [198.30131](#) in [80720.22](#) seconds. [Link](#) 
- Authors** Unknown – no solver description submitted  
**Affiliation** Unknown – no solver description submitted
- Rank 20 [lmsrusso](#) got a score of [198.19412](#) in [77464.5](#) seconds. [Link](#) 
- Authors** Luís M. S. Russo  
**Affiliation** INESC-ID and Department of Computer Science and Engineering, Instituto Superior Técnico, Universidade de Lisboa
- Rank 21 [GOAT](#) got a score of [196.02268](#) in [235.44](#) seconds. [Link](#) 
- Authors** Patrik Drbal, Michal Dvořák, Dušan Knop, Jozef Koleda, Jan Pokorný and Ondřej Suchý  
**Affiliation** Czech Technical University in Prague
- Rank 22 [NV\\_OCM](#) got a score of [194.69549](#) in [5519.86](#) seconds. [Link](#) 
- Authors** André Nusser and Juliette Vlieghe  
**Affiliation** CNRS, Inria Center at Université Côte d’Azur and Technical University of Denmark
- Rank 23 [HCPS42](#) got a score of [192.17673](#) in [2172.91](#) seconds. [Link](#) 
- Authors** Temirkhan Zimanov  
**Affiliation** Higher School of Economics
- Rank 24 [asdf](#) got a score of [188.77488](#) in [27596.93](#) seconds. [Link](#) 
- Authors** Unknown – no solver description submitted  
**Affiliation** Unknown – no solver description submitted
- Rank 25 [simonhol](#) got a score of [181.14588](#) in [4036.8](#) seconds. [Link](#) 
- Authors** Simon Hol  
**Affiliation** Utrecht University

## 26:14 PACE 2024: One-Sided Crossing Minimization

- Rank 26** [U\\_OCM](#) got a score of [181.05501](#) in [80636.53](#) seconds. [Link](#) 
- Authors** Mert Biyikli, Kathrin Hanauer, Sophia Heck, Lukas Krumpeck, Lara Ost, Tobias Prisching, Ole Schlüter, Matej Vedak and Maximilian Vötsch  
**Affiliation** Faculty of Computer Science, University of Vienna and Faculty of Physics, University of Vienna
- Rank 27** [iitg](#) got a score of [180.18229](#) in [480.38](#) seconds. [Link](#) 
- Authors** Sahaj Gupta, Swati Nanda Gupta, Sampriti Patel and Srinibas Swain  
**Affiliation** Department of Computer Science and Engineering, IIIT Guwahati
- Rank 28** [Guilucand](#) got a score of [172.30134](#) in [16750.69](#) seconds. [Link](#) 
- Authors** Andrea Cracco  
**Affiliation** Università degli Studi di Verona
- Rank 29** [DumbAndDumber](#) got a score of [165.28777](#) in [1904.8](#) seconds. [Link](#) 
- Authors** Kristoffer Sandvang and Mateusz Filipowski  
**Affiliation** Student at the University of Copenhagen
- Rank 30** [roundabout](#) got a score of [163.44052](#) in [32077.4](#) seconds. [Link](#) 
- Authors** Emmanuel Arrighi and Petra Wolf  
**Affiliation** EnsL, Université Lyon, UCBL, CNRS, Inria in Lyon and LaBRI, CNRS, Université de Bordeaux
- Rank 31** [oscmpp](#) got a score of [112.20293](#) in [54079.36](#) seconds. [Link](#) 
- Authors** Sahaj Gupta, Swati Nanda Gupta, Sampriti Patel and Srinibas Swain  
**Affiliation** Department of Computer Science and Engineering, IIIT Guwahati
- Rank 32** [WINTER](#) got a score of [109.28239](#) in [57113.3](#) seconds. [Link](#) 
- Authors** Sahaj Gupta, Swati Nanda Gupta, Sampriti Patel and Srinibas Swain  
**Affiliation** Department of Computer Science and Engineering, IIIT Guwahati

### 4.4 Methods used by the Winners of the Heuristic Track

The **Parameterized Track** winner **CIMAT\_Team** [36] used a first generation memetic algorithm with explicit diversity management. They initially generated a population of random solutions. They used iterated local search to improve these solutions. Their approach applied cycle-based crossover and applied a Best-Non-Penalized survivor selection strategy, which adjusts over time to favor exploration early and exploitation later, thereby promoting diversity. For larger instances, their method shifted to a more direct application of iterated local search. They used a greedy initialization strategy based on scoring vertices to manage complexity and memory efficiently.

Similar to team **uzl** in the exact track, the **runner-up LUNCH** [29] reduced OSCM to the **WEIGHTED FEEDBACK ARC SET** problem. They observed that edges between strongly connected components can be deleted as they are not part of any cycles, and that strongly connected components can be solved independently. They proved that several edges can be ignored and thus managed to create sparser instances that are sufficient to quickly find strongly connected components. They used a dynamic program [30] to solve components of at most 20 vertices optimally. For larger components, they first used greedy improvements, then an adjusted cutting technique by Park and Akers [34].

The **third-place Martin\_J\_Geiger** [16] used iterated local search and variable neighborhood search to find good solutions. They first applied the reduction rules by Dujmović et al. [12] and then used the barycenter heuristic to find an initial solution. They iteratively improved the solution with small improving moves until they reached a local optimum; to

this end, they either moved a single node or a block of up to 5 subsequent nodes to different positions. To escape local optima, they reversed a subset of up to 20% of the permutation and continued their search from there.

#### 4.5 Ranking of the Parameterized Track

The ranking for the parameterized track is listed subsequently; We list the number of solved instances from the 100 private instances plus the 100 public instances as well as the total computation time. Three teams were disqualified because instances were found for which their solver did return a suboptimal solution. Submissions marked with an “🎓” icon are student submissions after the same rules as above. There were two such submissions, which received the Student Submission Award.

- Rank 1** [LUNCH](#) solved 200 instances in 5.15 seconds. [Link](#) 🎓  
**Authors** Kenneth Langedal, Matthias Bentert, Thorgal Blanco and Pål Grønås Drange  
**Affiliation** University of Bergen
- Rank 1** [mjdv](#) solved 200 instances in 10.37 seconds. [Link](#) 🎓  
**Authors** Ragnar Groot Koerkamp and Mees de Vries  
**Affiliation** ETH Zurich and Unaffiliated in The Netherlands
- Rank 3** [mjpeg](#) solved 200 instances in 25.22 seconds. [Link](#) 🎓  
**Authors** Michael Jünger, Paul Jünger, Petra Mutzel and Gerhard Reinelt  
**Affiliation** University of Cologne, University of Bonn and Heidelberg University
- Rank 4** [Arcee](#) (🎓) solved 200 instances in 28.54 seconds. [Link](#) 🎓  
**Authors** Kimon Boehmer, Lukas Lee George, Fanny Hauser and Jesse Palarus  
**Affiliation** Université Paris-Saclay, Technical University Berlin
- Rank 5** [crossy](#) (🎓) solved 200 instances in 34.98 seconds. [Link](#) 🎓  
**Authors** Tobias Röhr and Kirill Simonov  
**Affiliation** Hasso Plattner Institute, University of Potsdam
- Rank 6** [uzl](#) solved 200 instances in 60.49 seconds. [Link](#) 🎓  
**Authors** Max Bannach, Florian Chudigiewitsch, Kim-Manuel Klein and Marcel Wienöbst  
**Affiliation** European Space Agency and Institute for Theoretical Computer Science, University of Lübeck
- Rank 7** [roundabout](#) solved 200 instances in 121.23 seconds. [Link](#) 🚫  
**Authors** Emmanuel Arrighi and Petra Wolf  
**Affiliation** EnsL, Université Lyon, UCBL, CNRS, Inria in Lyon and LaBRI, CNRS, Université de Bordeaux
- Rank 8** [CRGone](#) solved 200 instances in 125.07 seconds. [Link](#) 🎓  
**Authors** Alexander Dobler  
**Affiliation** Technische Universität Wien
- Rank 9** [Guilucand](#) solved 200 instances in 162.07 seconds. [Link](#) 🎓  
**Authors** Andrea Cracco  
**Affiliation** Università degli Studi di Verona
- Rank 10** [weberknecht](#) solved 200 instances in 287.41 seconds. [Link](#) 🎓  
**Authors** Johannes Rauch  
**Affiliation** Institute of Optimization and Operations Research, Ulm University
- Rank 11** [sherby](#) solved 199 instances in 20.84 seconds. [Link](#) 🎓  
**Authors** Manuel Lafond, Alitzel López Sánchez and Bertrand Marchand  
**Affiliation** Department of Computer Science, University of Sherbrooke



## 26:16 PACE 2024: One-Sided Crossing Minimization

**Rank 12** [trex-ufmg](#) solved **198** instances in **14848** seconds. [Link](#) 

**Authors** Luis Henrique Gomes Higino, Kaio Henrique Masse Vieira, Alan Prado, Guilherme de Castro Mendes Gomes, Laila Melo Vaz Lopes, Gabriel Ubiratan Barreto Pereira de Oliveira, Gabriel Lucas Costa Martins, Heitor Gonçalves Leite, Matheus Torres Prates, Gabriel Vieira and Vinicius Fernandes dos Santos

**Affiliation** Departamento de Ciência da Computação, Universidade Federal de Minas Gerais

**Rank 13** [narekb95](#) solved **163** instances in **13082.35** seconds. [Link](#) 

**Authors** Narek Bojikian

**Affiliation** Humboldt University of Berlin

### 4.6 Methods used by the Winners of the Parameterized Track

The **Parameterized Track winner LUNCH** [29] started with the same approach as in their second place submission to the heuristic track. After running the heuristic for each large component to get an upper bound, they solved a MaxSAT instance to optimally eliminate all cycles of length at most 4. If this led to an acyclic instances or to a solution that had the same cost as the upper bound, the algorithm terminated. Otherwise, they removed these edges, found new cycles with a DFS traversal, and repeated the previous steps.

The **runner-up mjdv** [27] did not rely on ILP or SAT formulations and instead used a branch-and-bound algorithm. They generalized reduction rules from Dujmović et al. [12] to find pairs of vertices in  $B$  where one must lie to the left of the other in any optimal solution, or pairs that are placed directly next to each other in some optimal solution. They also generalized a reduction rule to find a vertex that lies at the leftmost position in some optimal solution. As a lower bound, they calculated for each pair of vertices in  $B$  the minimum number of crossings required between their incident edges in any drawing [11, 15]. They then fixed vertices in the solution from left to right, keeping track of the number of crossings involved with the fixed vertices. It uses the reduction rules explained above to fix pairs of vertices and find the next vertex to add to the prefix, which is then inserted at the optimal position. Caching previously found lower bounds speeds up the process.

The **third-place mpeg** [24] used the same solver as in the exact track.

## 5 PACE Organization

The program committee of PACE 2024 consisted of Philipp Kindermann (Universität Trier, chair), Fabian Klute (UPC Barcelona) and Soeren Terziadis (Eindhoven University of Technology). During the competition, the members of the steering committee were:

- (since 2023) Max Bannach (European Space Agency)
- (since 2023) Sebastian Berndt (Universität zu Lübeck)
- (since 2016) Holger Dell (Goethe University Frankfurt and IT University of Copenhagen)
- (since 2016) Bart M. P. Jansen (chair) (Eindhoven University of Technology)
- (since 2020) Lukasz Kowalik (University of Warsaw)
- (since 2021) André Nichterlein (Technical University of Berlin)
- (since 2022) Christian Schulz (Universität Heidelberg)
- (since 2020) Manuel Sorge (Technische Universität Wien)

The Program Committee of PACE 2025 will be chaired by Sebastian Siebertz und Mario Grobler (both University of Bremen).

## 6 Conclusion, Reflection and Future Editions of PACE

We thank all the participants for their impressive work, vital contributions, and patience in case of technical issues. The organizers especially thank the participants who presented their work at IPEC 2024 in the poster session or during the award ceremony. We are pleased about the large number of diverse participants and hope the PACE challenge will continue to build bridges between theory and practice. We welcome anyone interested to add their name to the mailing list on the PACE website to receive updates and join the discussion.

### The Good – What went well

OSCM was a good choice as the problem for the competition: the problem is simple to understand, requires no background knowledge, and the solutions are easy to verify. We used a Zulip server for direct communication with and between the participants. This server was very active, it made it much easier for us to communicate updates with the participants, and gave them the opportunity to discuss issues and help each other. We had a very large number of participants, much more than in the previous years. While many of the approaches were similar in nature, each team found different tricks to reduce instance sizes or to overcome obstacles. The poster exhibition during IPEC where the winning teams shared the knowledge they obtained about the problem during the contest was well attended and sparked several hours worth of discussion. Overall, there was a very friendly and helpful atmosphere between contestants. The exact and heuristic tracks were very successful with many different submissions. The participants were also very quick in finding any technical mistakes that the program committee had made in the provided tools and instances.

### The Bad – What could we have done better

Since we guaranteed the exact instances to have not too large crossing numbers, we could only use instances for the benchmark sets where we knew the optimum solution. But since our internal solver turned out to be very slow compared to the submissions we received, this limited the difficulty for the instances we could provide. For example, our solver needed in total 567 hours to solve the 200 instances of the whole parameterized benchmark set, while the best submission could solve all of them in merely 5 seconds. In the future, we suggest that there should also be graphs where the optimum solution is unknown, even if that means that one cannot give guarantees on the solution. It might make sense to follow the example of the SAT competition, where each participating team also has to submit a small number of interesting benchmark instances that are then added to the evaluation set.

Because of our slow solver, we severely overestimated the difficulty of the parameterized benchmark set. Hence, the parameterized track was more akin to an algorithm engineering competition where the participants were fighting to scrape off milliseconds from their running times instead of figuring out how to solve larger instances. To the best of our knowledge, no submission in this track even used the property that the graphs have small cutwidth. Overall, the parameterized track did not work out well this time and should be strongly revised before running it again; there should at least be much larger graphs, larger parameters and/or less allowed computation time.

The public repository that we set up for the community to share interesting instances with each other was unused – there have been no pushes except by the program committee. After the final evaluation and publishing the private benchmark sets, one participant found that 5 of the instances had multi-edges. All input graphs were supposed to be simple; this

was due to a bug in our graph generators. We had to fix these instances, rerun all solvers on the fixed versions and change the ranking after publishing the results, which was unfortunate for teams that dropped in the rankings.

### **The Ugly – What did we struggle with**

We struggled a bit with the tight timeline, as we underestimated the huge computation times required to find solutions for the benchmark sets. We had access to a 92-core server that was running non-stop for 4 months. It was very tough for us to predict how hard instances are and to find interesting instances that fit all constraints to make them interesting: for example, they should have not too few edges, not require too many crossings, and common heuristics should not immediately find optimum solutions. Many of these could only be checked after finding their optimum solution. Thus, we had to create a huge number of potential instances, many of which turned out to be unusable.

The servers by Optil.io were overloaded by the many submissions. During the last few days, participants had to wait for many hours until they received a result from their submitted solvers. This also had the effect that the running times varied a lot; submitting the same solver twice could lead to completely different timing results. Unfortunately, we could not find an alternative, as the required server load is too large for non-commercial options. Hence, we had to evaluate all solvers on our own server after the submission deadline. This also took a long time: to ensure that each run receives the exact same resources, we could only solve 20 instances in parallel, each of which could take a bit more than 30 minutes, so we needed up to 5 hours computation for each each of the 75 submissions. While some solvers were easy to compile and run, others required more time: 15 solvers did not immediately compile or run without major issues and required help from the teams. To reduce the workload for the program committee in the future, one option would be to let the participants submit docker images that should remove these issues.

Finally, one large point of discussion were the rules for the exact track, in particular that it was not a formal requirement that submissions have to be based on provably optimal algorithms. The reason for this non-requirement was to encourage new techniques even without finding a theoretical proof for them. However, this led to two submissions that were based on heuristic solvers that just happened to solve also all instances of the exact benchmark set optimally. The submissions were disqualified due to subsequently found counterexample instances. For the future, these rules should be revised; either there should be more strict requirements or the community should get the option to review submitted solvers and find counterexamples before the rankings are published.

### **PACE 2025**

We look forward to the next edition, which will focus on **DOMINATING SET** and **HITTING SET** and will be chaired by Sebastian Siebertz and Mario Grobler. Detailed information will be posted on the website of the competition at [pacechallenge.org](http://pacechallenge.org).

---

### **References**

- 1 Ali Baharev, Hermann Schichl, Arnold Neumaier, and Tobias Achterberg. An exact method for the minimum feedback arc set problem. *ACM J. Exp. Algorithmics*, 26, April 2021. doi:10.1145/3446429.
- 2 Tomas Balyo, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors. *SAT Competition 2023: Solver, Benchmark and Proof Checker Descriptions*. Uni. Helsinki, 2023.

- 3 Max Bannach and Sebastian Berndt. PACE solver description: The PACE 2023 parameterized algorithms and computational experiments challenge: Twinwidth. In *IPEC*, volume 285 of *LIPICs*, pages 35:1–35:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.IPEC.2023.35.
- 4 Max Bannach, Florian Chudigiewitsch, Kim-Manuel Klein, and Marcel Wienöbst. PACE solver description: UzL exact solver for one-sided crossing minimization. In *IPEC*, volume 321 of *LIPICs*, pages 28:1–28:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 5 Édouard Bonnet and Florian Sikora. The PACE 2018 parameterized algorithms and computational experiments challenge: The third iteration. In *IPEC*, volume 115 of *LIPICs*, pages 26:1–26:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.IPEC.2018.26.
- 6 Holger Dell, Thore Husfeldt, Bart M. P. Jansen, Petteri Kaski, Christian Komusiewicz, and Frances A. Rosamond. The first parameterized algorithms and computational experiments challenge. In *IPEC*, volume 63 of *LIPICs*, pages 30:1–30:9. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.IPEC.2016.30.
- 7 Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration. In *IPEC*, volume 89 of *LIPICs*, pages 30:1–30:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.IPEC.2017.30.
- 8 Hristo N. Djidjev and Imrich Vrto. Crossing numbers and cutwidths. *J. Graph Algorithms Appl.*, 7(3):245–251, 2003. doi:10.7155/JGAA.00069.
- 9 Alexander Dobler. A note on the complexity of one-sided crossing minimization of trees, 2023. arXiv:2306.15339, doi:10.48550/arXiv.2306.15339.
- 10 Alexander Dobler. PACE solver description: CRGone. In *IPEC*, volume 321 of *LIPICs*, pages 29:1–29:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 11 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.
- 12 Vida Dujmović, Henning Fernau, and Michael Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. *J. Discr. Alg.*, 6(2):313–323, 2008. doi:10.1016/j.jda.2006.12.008.
- 13 M. Ayaz Dzulfikar, Johannes Klaus Fichte, and Markus Hecher. The PACE 2019 parameterized algorithms and computational experiments challenge: The fourth iteration (invited paper). In *IPEC*, volume 148 of *LIPICs*, pages 25:1–25:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.IPEC.2019.25.
- 14 Peter Eades and Sue Whitesides. Drawing graphs in two layers. *Theoret. Comp. Sci.*, 131(2):361–374, 1994. doi:10.1016/0304-3975(94)90179-1.
- 15 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. doi:10.1007/bf01187020.
- 16 Martin J. Geiger. PACE 2024 solver description: Martin\_J\_Geiger. In *IPEC*, volume 321 of *LIPICs*, pages 32:1–32:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 17 Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: Obstructions and algorithmic aspects. *Algorithmica*, 81(2):557–588, 2019. doi:10.1007/S00453-018-0424-7.
- 18 Ernestine Großmann, Tobias Heuer, Christian Schulz, and Darren Strash. The PACE 2022 parameterized algorithms and computational experiments challenge: Directed feedback vertex set. In *IPEC*, volume 249 of *LIPICs*, pages 26:1–26:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.IPEC.2022.26.
- 19 Martin Grötschel, Michael Jünger, and Gerhard Reinelt. A cutting plane algorithm for the linear ordering problem. *Oper. Res.*, 32(6):1195–1220, 1984. doi:10.1287/OPRE.32.6.1195.
- 20 Martin Grötschel, Michael Jünger, and Gerhard Reinelt. Facets of the linear ordering polytope. *Math. Program.*, 33(1):43–60, 1985. doi:10.1007/BF01582010.

- 21 Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *SciPy*, pages 11–15, 2008.
- 22 Patrick Healy and Nikola S. Nikolov. Hierarchical drawing algorithms. In *Handbook on Graph Drawing and Visualization*, pages 409–453. Chapman and Hall/CRC, 2013.
- 23 Michael Jünger and Petra Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. In *J. Graph Algorithms Appl.*, volume 1, pages 1–25. World Scientific, 2002. doi:10.1142/9789812777638\_0001.
- 24 Michael Jünger, Paul Jünger, Petra Mutzel, and Gerhard Reinelt. PACE solver description: Exact solution of the one-sided crossing minimization problem by the MPPEG team. In *IPEC*, volume 321 of *LIPICs*, pages 27:1–27:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 25 Leon Kellerhals, Tomohiro Koana, André Nichterlein, and Philipp Zschoche. The PACE 2021 parameterized algorithms and computational experiments challenge: Cluster editing. In *IPEC*, volume 214 of *LIPICs*, pages 26:1–26:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.IPEC.2021.26.
- 26 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.
- 27 Ragnar Groot Koerkamp and Mees de Vries. PACE solver description: OCMu64, a solver for one-sided crossing minimization. In *IPEC*, volume 321 of *LIPICs*, pages 35:1–35:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 28 Lukasz Kowalik, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Manuel Sorge, and Piotr Wygocki. The PACE 2020 parameterized algorithms and computational experiments challenge: Treedepth. In *IPEC*, volume 180 of *LIPICs*, pages 37:1–37:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.IPEC.2020.37.
- 29 Kenneth Langedal, Matthias Bentert, Thorgal Blanco, and Pål Grønås Drange. PACE solver description: LUNCH — linear uncrossing heuristics. In *IPEC*, volume 321 of *LIPICs*, pages 34:1–34:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 30 E. Lawler. A comment on minimum feedback arc sets. *IEEE Trans. Circuit Theory*, 11(2):296–297, 1964. doi:10.1109/TCT.1964.1082291.
- 31 Xiao Yu Li and Matthias F. Stallmann. New bounds on the barycenter heuristic for bipartite drawing. *Inform. Proc. Let.*, 82:293–298, 2002. doi:10.1016/S0020-0190(01)00297-6.
- 32 Xavier Muñoz, Walter Unger, and Imrich Vrto. One sided crossing minimization is NP-hard for sparse graphs. In *GD*, volume 2265 of *LNCS*, pages 115–123. Springer, 2001. doi:10.1007/3-540-45848-4\_10.
- 33 Networks project, 2017. URL: <https://www.thenetworkcenter.nl>.
- 34 S. Park and S.B. Akers. An efficient method for finding a minimal feedback arc set in directed graphs. In *ISCAS*, volume 4, pages 1863–1866, 1992. doi:10.1109/ISCAS.1992.230449.
- 35 Martí Rafael and Reinelt Gerhard. Exact and heuristic methods in combinatorial optimization. *Appl. Math. Sci.*, 2022. doi:10.1007/978-3-662-64877-3.
- 36 Carlos Segura, Lázaro Lugo, Gara Miranda, and Edison David Serrano Cárdenas. PACE solver description: CIMAT\_Team. In *IPEC*, volume 321 of *LIPICs*, pages 31:1–31:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 37 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiro Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst., Man, and Cybernetics*, 11(2):109–125, 1981. doi:10.1109/tsmc.1981.4308636.
- 38 Markus Wallinger. *Exploring Graph-based Concepts for Balanced Information Density in Data Visualizations*. PhD thesis, TU Wien, 2024. doi:10.34726/hss.2024.124826.