



PACE Solver Description: The PACE 2023 Parameterized Algorithms and Computational Experiments Challenge: Twinwidth

Max Bannach  

European Space Agency, Advanced Concepts Team, Noordwijk, The Netherlands

Sebastian Berndt  

Institute for Theoretical Computer Science, University of Lübeck, Germany

Abstract

This article is a report by the challenge organizers on the 8th Parameterized Algorithms and Computational Experiments Challenge (PACE 2023). 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, the problem was to compute the *twinwidth* of a graph, a recently introduced width parameter that measures the similarity of a graph to a cograph. 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. The same task must be accomplished within 5 minutes in the heuristic track. However, the result in this track is not required to be optimal.

As in previous iterations, the organizers handed out awards to the best solutions in both tracks and to the best student submissions. New this year is a dedicated *theory award* that appreciates new theoretical insights found by the participants during the development of their tools.

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

Keywords and phrases Twinwidth, Algorithm Engineering, FPT, Kernelization

Digital Object Identifier 10.4230/LIPIcs.IPEC.2023.35

Acknowledgements The prize money (€4000) was generously provided by Networks [28], 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, led by Szymon Wasik, and especially to Jan Badura and Artur Laskowski for the fruitful collaboration and for hosting the competition at the optil.io online judge system. We also thank André Schidler and Stefan Szeider, who made their exact solver available to the organizers prior to the competition for internal evaluations [29].

1 Introduction: History and Timeline of PACE

The Parameterized Algorithms and Computational Experiments Challenge (PACE) is an algorithm engineering competition conceived in 2015 and held annually since. Its aim is to bridge the theory of parameterized algorithms and their use in practice. The goals are to:

1. bridge the divide between the theory of algorithm design and analysis, and the practice of algorithm engineering,
2. inspire new theoretical developments,
3. investigate the competitiveness of theoretical algorithms from the field of parameterized complexity analysis and related fields in practice,
4. produce universally accessible libraries of implementations and repositories of benchmark instances, and
5. encourage the dissemination of these findings in scientific papers.



© Max Bannach and Sebastian Berndt;

licensed under Creative Commons License CC-BY 4.0

18th International Symposium on Parameterized and Exact Computation (IPEC 2023).

Editors: Neeldhara Misra and Magnus Wahlström; Article No. 35; pp. 35:1–35:14

Leibniz International Proceedings in Informatics



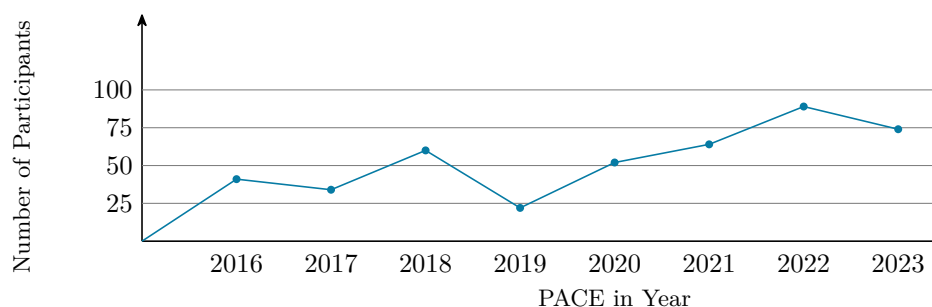
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

35:2 PACE Solver Description: PACE 2023: Twinwidth

In each of the now eight iterations, the participants were asked to provide implementations that either solved small- to medium-sized instances of an NP-complete problem optimally or to solve large-sized instances approximately. In previous iterations, the problems were

PACE 2016	TREEWIDTH and UNDIRECTED-FEEDBACK-VERTEX-SET	[14];
PACE 2017	TREEWIDTH and MINIMUM FILL-IN	[15];
PACE 2018	STEINER-TREE	[12];
PACE 2019	VERTEX-COVER and HYPERTREEWIDTH	[17];
PACE 2020	TREEDEPTH	[22];
PACE 2021	CLUSTER-EDITING	[21];
PACE 2022	DIRECTED-FEEDBACK-VERTEX-SET	[18].

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. Over the years, the number of PACE participants has constantly grown, as Figure 1 illustrates. Furthermore, papers inspired by concrete implementations 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 [5].

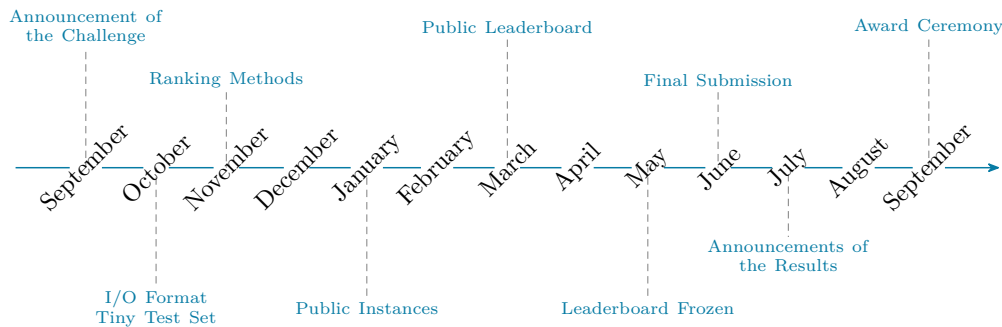


■ **Figure 1** Overview of the number of participants (y -axis) of the PACE challenge over the years.

In this article, we report on the eighth iteration of the PACE challenge. The problem chosen for this year's iteration was *twinwidth*, a relatively young but promising parameter introduced in 2020 by Bonnet et al. [10]. The challenge featured two tracks: an exact track and a heuristic 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. In the heuristic track, the task was to compute a valid solution with a width as small as possible within a time limit of 5 minutes and a memory limit of 8 GB.

The PACE 2023 challenge was announced with both tracks in September 2022. Details about the input and output format were provided in October 2022 together with a tiny test set to allow the participants to start with the challenge. One reason being, in particular, that algorithmic lectures that eventually start in this period may integrate PACE into course related projects. The concrete ranking methods for both tracks were published in November 2022. In January 2023, the public instances were made available to the participants, and the public leaderboard on the `optil.io` platform was opened in March 2023. This allowed the participants to test their solvers on the public instances and provided a provisional ranking. The leaderboard was frozen in May 2023, and the final version of the submission was due on the first of June 2023. Afterwards, the submissions were evaluated on the private

instances, which the participants did not know. The results of this evaluation were announced in July 2023, and the award ceremony took place during the International Symposium on Parameterized and Exact Computation (IPEC) 2023 in Amsterdam in September 2023. The complete timeline can be found in Figure 2.



■ **Figure 2** Timeline of the PACE challenge in 2023 (the diagram ranges from September 2022 to September 2023). The next iteration of the PACE for 2024 was announced at the award ceremony.

2 The Problem of the Challenge: Twinwidth

So-called *width measures* are graph parameters that capture the structural complexity of a graph in terms of the minimum cost of an associated type of decomposition of the graph. The poster child of such a parameter is *treewidth*, which measures the distance of a graph to a tree. A recently proposed parameter is *twinwidth*, introduced by Bonnet, Kim, Thomassé, and Watrigant [11]. Informally, the twinwidth of a graph measures the distance to a *cograph*. Cographs are the graphs that can be generated from a single-vertex graph by complementation and disjoint union; which are precisely the P_4 -free graphs. In the context of twinwidth, the crucial property of these graphs is that they always contain a pair of *twins*, that is, vertices sharing the same neighborhood. The contraction of twins in a cograph results again in a cograph. In other words, cographs can be reduced to a single vertex by a sequence of twin contractions.

If the input graph $G = (V, E)$ is not a cograph, this process of contracting twins will eventually get stuck with a graph that contains more than one vertex, but does not contain any twins. We consider the contraction of non-twins (which may or may not be adjacent) as “error” (because the goal is to measure the distance to cographs) and record this error by coloring mixed edges “red,” while referring to the original edges as “black edges”. Precisely, when contracting two vertices u and v into a single fresh vertex z , we form a new graph by removing u and v . For each $x \in V \setminus \{u, v\}$ that was connected to u or v , we add a new edge between x and z . If x was connected to both u and v and both of these edges were black, the edge between x and z is black as well. In all other cases, the edge between x and z is colored red. Figure 3 presents an example: When a and c are contracted in the second picture into the fresh vertex ac , then d gets connected to ac with a black edge (as d was connected to both a and c). In contrast, e was only connected to c but not to a and, hence, a red edge now connects e and ac . Finally, b was not connected to a or to c and hence, no edge is drawn between b and ac .

A *contraction sequence* consists of $|V| - 1$ contractions transforming the input graph (which does not contain any red edges) into a single vertex (which also does not contain red edges). However, intermediate graphs in the sequence may contain red edges. The *width* of a

contraction sequence is the maximal red degree of any vertex encountered in any intermediate graph. The *twinwidth* of a graph G is the minimal width any contraction sequence of G must have. We denote it by $\text{tw}(G)$ and derive the following computational task:

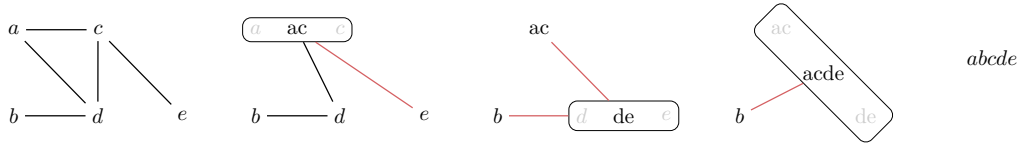
► **Challenge Problem** (TWINWIDTH).

Instance: A undirected graph G given as list of edges.

Task: Compute a contraction sequence S of G and output it as list of pairs.

Exact Track: The width of S must equal $\text{tw}(G)$.

Heuristic Track: Any valid sequence S can be output. The quality is the width of S .



■ **Figure 3** An exemplary contraction sequence of width 2 from the The sequence shows that the input graph (most left) has twinwidth at most 2. The reason being that the third graph has a red degree of 2.

Twinwidth has received wide attention in the three years following its inception: dblp alone lists (at the time of writing this report) 4 papers related to twinwidth in 2020, 15 in 2021, and already 37 in 2022. One of the reasons for its success is that the class of graphs of bounded twinwidth is one of the largest known graph classes that still allows first-order model checking in fpt-time [11]. There is also a growing number of intractable problems for which efficient algorithms on graphs of bounded twinwidth were found [8, 10]. However, all these results require that a contraction sequence of small width is given along with the input, i. e., one needs to solve the aforementioned TWINWIDTH problem as a preprocessing step. Most desirable would, of course, be an algorithm with polynomial running time that computes a contraction sequence of minimal width, or at least of width $\alpha \text{tw}(G)$ for some constant α . Unfortunately, the existence of an exact XP-algorithm or an approximation algorithm with $\alpha < 5/4$ was ruled out by Bergé, Bonnet, and Déprés, who proved that deciding whether $\text{tw}(G)$ is at most 4 is already NP-complete [7]. Nevertheless, nothing is known about $\alpha \geq 5/4$.

This is in strong contrast to problems chosen in previous iterations of PACE, where usually a portfolio of parameterized algorithms was available before the competition. That is, the first aim of PACE mentioned in the introduction, i. e., to *bridge the divide between the theory of algorithm design and analysis, and the practice of algorithm engineering* was interpreted mainly as “evolving theoretical insights into practical tools”. With TWINWIDTH as the problem of the challenge, the direction of this reasoning is inverted in this year’s iteration, i. e., we wondered whether “techniques commonly used in practice may lead to new results in theory”? This is also the reason that the organizers decided to hand out a dedicated theory award along with the usual ranking: To value and highlight theoretical insights gained during the competition in spirit of the second aim of the competition (to *inspire new theoretical developments*).

3 The Setup of PACE 2023

As already mentioned before, this year’s challenge featured two tracks: An *exact* track in which a contraction sequence of minimal width needs to be computed, and a *heuristic* track in which a not necessarily optimal contraction sequence must be computed very quickly.

3.1 The Exact Track (Compute an Optimal Contraction Sequence)

The task of this track is to compute an optimal solution of TWINWIDTH for 100 graphs, which are not known by the participants (they are only presented to the solver during the evaluation in a compartmentalized judge system). For each of the graphs, the solver has to output a solution within a time limit of *30 minutes* and a memory limit of *8 GB*.

The organizers of the competition encouraged submissions that implement provably optimal algorithms, however, this was not a formal requirement. Instead, submissions that output an incorrect solution or a solution that is known to be non-optimal were disqualified from the challenge. Fortunately, no submission was disqualified this year.

Submissions of this track are ranked by the *number of solved instances*. In case of a tie, the winner is 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 (Compute a Contraction Sequence Quickly)

In this track the solver shall compute a good solution quickly. The solver are run on each instance for *5 minutes* and receive the Unix signal SIGTERM afterwards. When receiving this signal, the solver has to output a correct contraction sequence immediately to the standard output and terminate. If the program does not halt in a reasonable time after receiving the signal, it will be stopped via SIGKILL and the instance is counted as time limited exceeded. The memory limit for this track is *8 GB* as well.

For this track solutions do not have to be optimal. However, solvers that produce incorrect solution were disqualified. Fortunately, we did not need to disqualify any solver in this track either. Submissions were ranked by the geometric mean over all instances of

$$100 \cdot \frac{\text{width produced by the solver}}{\text{smallest width known to the PC}}$$

Note that the “smallest width known to the PC” may not be optimal, i. e., may be larger than $\text{tww}(G)$.

3.3 Internal Solver and the Benchmark Set

The fourth aim of the PACE challenge is to *produce universally accessible libraries of implementations and repositories of benchmark instances*. While the first part of this aim is exactly what we expect from the participants, it is the duty of the program committee to produce the benchmark instances. A lot can be said about how to set up a good benchmarkset, and what “good” in this context actually means. In the light of PACE, we found the following points important:

- a) the benchmarkset contains instances from various domains,
- b) it contains easy, medium, hard instances,
- c) it remains a challenging benchmark after the challenge.

The reason for a) being that we would like to evaluate the overall performance of the approaches developed by the participants (and not the performance on, say, a specific graph class). We include b) to make the challenge interesting and fun. We wanted a benchmarkset in which every participant can solve at least a few instances, which should especially encourage student teams to participate as well. The medium instances are the ones that are meant to distinguish the quality of the various solvers, and the hard instances realize c). This last

item is included, as we did not want a benchmarkset that is “completely 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.

The emerging question of course is: “What is an easy, medium, or hard instance”? This question is not easy to answer, especially if barely anything is known about solving the problem. The organizers used a SAT-based internal solver to answer this question (the solver was developed by André Schidler and Stefan Szeider [30], who provided the organizers with an improved version of the tool before the competition).

The *base set* of graphs was generated from three sources:

Random and synthetic graphs, including: Erdős-Rényi graphs, random hyperbolic graphs, random planar graphs, and graph products of various simple graph classes.

Instances from graph repositories, including: road networks, power grid graphs, graphs from the UAI competition 2014, and max-flow instances from applications in computer vision.

Instances generated by reduction from SAT, including: the instances of the SAT competition 2022 *Anniversary Track*, reduced to VERTEX-COVER and 3-COLORING.

From this base set we generated an *enhanced set* by performing the following steps to every graph multiple times: (i) sample a random vertex, (ii) perform a BFS for ℓ layers to obtain a smaller subgraph, and (iii) add some noise to this subgraph by randomly (with a small probability ϵ) add, remove, or contract edges. By varying ℓ and ϵ , graphs of various sizes and difficulties can be generated. We ended up with an enhanced set of roughly 8000 graphs. To obtain the benchmarkset used in the competition, we let the internal solver run on every instance of the enhanced set for eight hours. We classified an instance as

Easy if the internal solver solved the instance in *30 minutes*;

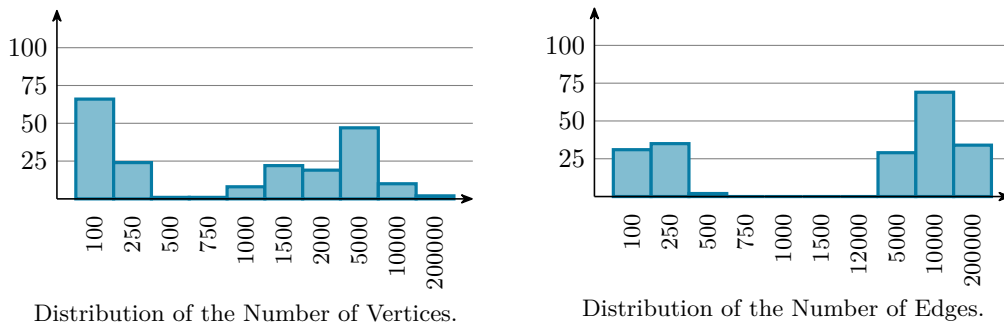
Medium if the internal solver *solved* the instance;

Hard if the internal solver did *not* solve the instance.

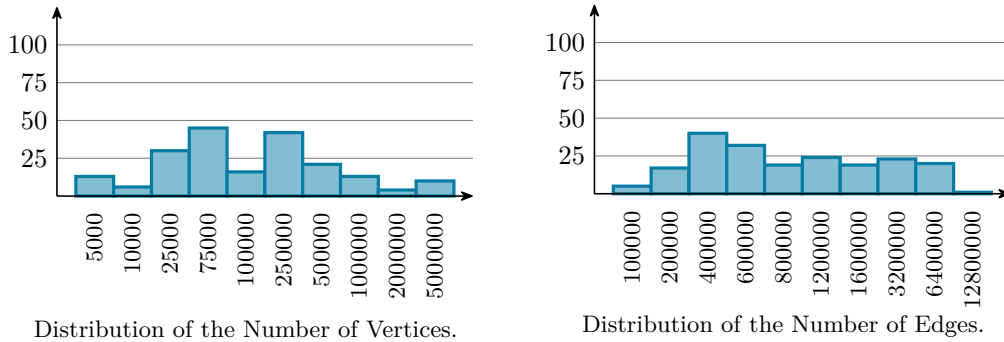
For the *exact track* we sampled 200 instances from this set (50 easy, 50 medium, and 100 hard). We sorted these 200 instances by size. Graphs with an odd number were used as private set to rank the participants, graphs with an even number were released in advance as public test set. Figure 4 illustrates the number of vertices (left) and number of edges (right) that the 200 instances of the *exact track* have. We used the same technique to derive the instances for the heuristic track, but scaled them by varying (the aforementioned parameter) ℓ after the classification into easy, medium, and hard. Figure 5 shows the distribution of the number of vertices and edges of the benchmarkset from the heuristic track. Of course, in the light of parameterized algorithms, not just the size of the instance, but also the actual *twinwidth* is important. All instances from the exact track that were solved during the competition have a twinwidth of at most 10 (see Figure 6 for the distribution). The instances used in the heuristic track have a larger twinwidth, but over 50% of the instances have a twinwidth of at most 60 (see the right plot in Figure 6). Both benchmarksets (all 200 instances per track) are publicly available at <https://pacechallenge.org/2023/>.

4 Participants and Results

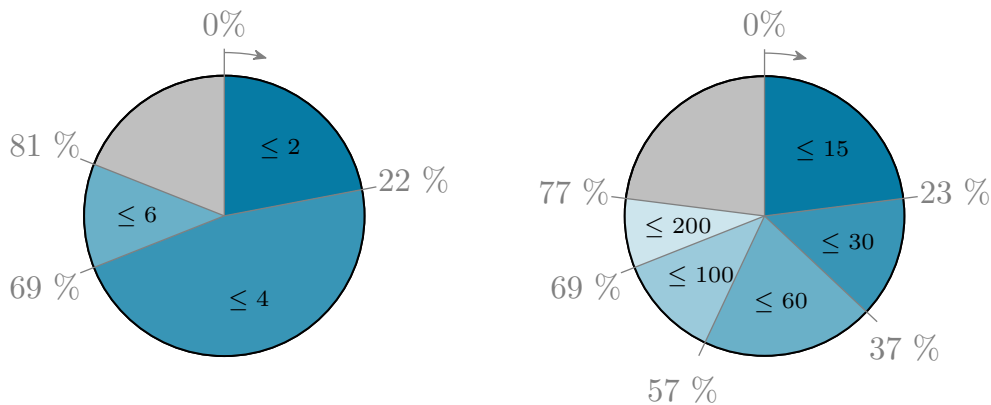
There were 13 and 10 teams that officially submitted a solution to the exact and heuristic track, respectively. There were four teams that submitted solutions to both tracks. Hence, in total there were 19 distinct teams with a total number of 73 participants representing three continents and a wide range of countries, which made this the second largest PACE challenge. The results are listed below.



■ **Figure 4** Distribution of the number of vertices (left) and number of edges (right) of the 200 instances of the PACE 2023 *exact* track. A bar at index x illustrates the number of instances that have at most x vertices (edges), but more than the previous bar contained. That is, every instance is counted in exactly one bar.




■ **Figure 5** Distribution of the number of vertices (left) and number of edges (right) of the 200 instances of the PACE 2023 *heuristic* track. A bar at index x illustrates the number of instances that have at most x vertices (edges), but more than the previous bar contained. That is, every instance is counted in exactly one bar.




■ **Figure 6** All private instances of the exact track that were solved by any solver have a twinwidth of at most 10. The left plot illustrates the twinwidth distribution these instances. The diagram shows clockwise the percentage of instances that fall below a certain twinwidth threshold. The right plot illustrates the same information for the private instances of the heuristic track (the best twinwidth found by any submission was assigned to every instance).

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. Submissions marked with “” 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).

The first three student submissions obtained a *Student Submission Award*. The submission marked with “” obtained the *Theory Award*, details follow in Section 4.5.

Rank 1 [Hydra Prime](#) solved 36 instances in 450 minutes.  [16]

Authors Yosuke Mizutani, David Dursteler, and Blair D. Sullivan

Affiliation University of Utah

Rank 2 [GUTHMi](#) solved 35 instances in 1257 minutes. [13]

Authors Alexander Leonhardt, Holger Dell, Anselm Haak, Frank Kammer, Johannes Meintrup, Ulrich Meyer, and Manuel Penschuck

Affiliation Goethe University Frankfurt and THM, University of Applied Sciences Mittelhessen

Rank 3 [Touiouidth](#) solved 34 instances in 8885 minutes. [9]

Authors Gaétan Berthe, Yoann Coudert–osmont, Alexander Dobler, Laure Morelle, Amadeus Reinald, and Mathis Rocton

Affiliation LIRMM, CNRS, Université de Montpellier and Université de Lorraine, CNRS, Inria and Algorithms and Complexity Group, TU Wien

Rank 4 [UAIC Twin Width](#) solved 34 instances in 12781 minutes.  [3]

Authors Andrei Arhire, Matei Chiriac, and Radu Timofte

Affiliation Alexandru Ioan Cuza University of Iasi, and ETH Zürich and University of Würzburg

Rank 5 [Soapen](#) solved 33 instances in 250 minutes. [33]

Authors Christopher Weyand and Marcus Wilhelm

Affiliation KIT Karlsruhe

Rank 6 [GBathie](#) solved 31 instances in 1183 minutes. [6]

Authors Gabriel Bathie, Jérôme Boillot, Nicolas Bousquet and Théo Pierron

Affiliation DI ENS, PSL Research University and LaBRI, Université de Bordeaux and Université Lyon

Rank 7 [Zygosity](#) solved 29 instances in 391 minutes. [23]

Authors Emmanuel Arrighi, Petra Wolf, Pål Grønås Drange, Kenneth Langedal, Farhad Vadiée, and Martin Vatselle

Affiliation University of Trier and University of Bergen

Rank 8 [trex-ufmg](#) solved 27 instances in 14760 minutes.  [32]

Authors Alan Cabral Trindade Prado, Emanuel Juliano Morais Silva, Guilherme de Castro Mendes Gomes, Kaio Henrique Masse Vieira and Laila Melo Vaz Lopes

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

- Rank 9** [tuw](#) solved 26 instances in 967 minutes. [30]
Authors André Schidler and Stefan Szeider
Affiliation Algorithms and Complexity Group, TU Wien
- Rank 10** [HeiTwin](#) solved 15 instances in 1187 minutes. 🏆 [26]
Authors Thomas Möller, Nikita-Nick Funk, Dennis Jakob and Ernestine Großmann
Affiliation Heidelberg University
- Rank 11** [A Simple Twin-width Searcher](#) solved 14 instances in 1490 minutes. [24]
Authors Alexander Meiburg
Affiliation University of California Santa Barbara
- Rank 12** [GOAT](#) solved 8 instances in 1487 minutes. [1]
Authors Adam Barla, Václav Blažej, Radovan Červený, Michal Dvořák, Dušan Knop, Josef Koleda, Jan Pokorný, Petr Štastný and Ondřej Suchý
Affiliation Faculty of Information Technology, Czech Technical University in Prague and University of Warwick
- Rank 13** [satwin](#) solved 8 instances in 35833 minutes. [20]
Authors Yinon Horev, Shiraz Shay, Sarel Cohen, Tobias Friedrich, Davis Issac, Lior Kamma, Aikaterini Niklanovits and Kirill Simonov
Affiliation School of Computer Science, The Academic College of Tel Aviv-Yaffo and Hasso Plattner Institute, University of Potsdam

4.2 Strategies Used in the Exact Track


We give a short overview on the ideas used by the top three solvers. The interested reader is referred to the corresponding papers published in the proceedings of IPEC 2023.

Winning Team. The winning solver, [Hydra Prime](#), first applies a modular decomposition on the graph and then considers all of the prime graphs and the quotient graph separately. Then, a vast number of different algorithms is applied to derive upper and lower bounds for the twinwidth of the graphs by choosing from four algorithms for the lower bounds and three algorithms for the upper bound. In addition, the solver also contains an optimal solver for trees, a branch-and-bound approach, and a SAT-based tool for the remaining cases. The main contributions are the *timeline encoding* and the *hydra decomposition*. The timeline encoding is a data structure that efficiently computes the width of a given contraction sequence by building up a set of red intervals representing the red edges during the contraction. The hydra decomposition is a refinement strategy that iteratively uses small vertex separators by splitting the graph into a set of heads and tails. This allows to find separators avoiding the heads, to contract the heads, and to join two hydras efficiently.

Runner-Up. The runner-up solver, [GUTHMI](#), is based on the branch-and-bound paradigm, but drastically reduces the potentially large branching vector using a number of heuristics. The authors obtain upper bounds using a heuristic, and lower bounds by sampling carefully chosen subgraphs that are tried to be solved exactly. Finally, to order the remaining branches, the authors use a scoring method to evaluate pairs of vertices. The scoring method aims to minimize the difference between the red degrees of vertices for which a new red edge will appear when contracting the pair, and the red degrees of vertices for which a red edge will be removed. The solver also contains several low-level optimizations such as the caching of small infeasible subproblems, the reusing of scores, and the use of appropriate data structures depending on the structure of the given graph.

Third Place. The third place solver, [Touiouidth](#), uses the first 15 minutes to find a good lower bound by considering the twinwidth of several carefully constructed induced subgraphs. Next, two different heuristics are run for five minutes to obtain upper bounds. Finally, the last ten minutes are used for a branch-and-bound algorithm. To reduce the search space, the authors use the observation that two vertices u and v can safely be contracted if the black neighborhood of u is a subset of the black neighborhood of v and, in addition, all neighbors (black and red) of v are contained in the red neighborhood of u (or are equal to u or v). To sort the possible branches for different pairs $\{u, v\}$, the authors first compute the size of the symmetric difference in the neighborhoods of u and v and then perform bucket-sort.

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, computed as described above. The larger the score, the better. As in the exact track, submissions marked with “” are student submissions of which the top three obtained a *Student Submission Award*.

Rank 1 [GUTHM](#) got a score of [9103](#) in [30004](#) seconds.  [13]

Authors Alexander Leonhardt, Holger Dell, Anselm Haak, Frank Kammer, Johannes Meintrup, Ulrich Meyer and Manuel Penschuck

Affiliation Goethe University Frankfurt and THM, University of Applied Sciences Mittelhessen

Rank 2 [Zygosity](#) got a score of [8603](#) in [24924](#) seconds. [23]

Authors Emmanuel Arrighi, Petra Wolf, Pål Grønås Drange, Kenneth Langedal, Farhad Vadiie, and Martin Vatschelle

Affiliation University of Trier and University of Bergen

Rank 3 [Red Alert](#) got a score of [7739](#) in [22924](#) seconds. [34]

Authors Édouard Bonnet and Julien Duron

Affiliation Université Lyon

Rank 4 [HATTER](#) got a score of [6169](#) in [30003](#) seconds.  [2]

Authors Aman Jain, Sachin Agarwal, Talika Gupta, and Srinibas Swain

Affiliation Department of Computer Science and Engineering, IIIT Guwahati

Rank 5 [HeiTwin](#) got a score of [6165](#) in [30122](#) seconds.  [27]

Authors Thomas Möller, Nikita-Nick Funk, Dennis Jakob, and Ernestine Großmann

Affiliation Heidelberg University

Rank 6 [UAIC Twin Width](#) got a score of [5937](#) in [28015](#) seconds.  [3]

Authors Andrei Arhire, Matei Chiriac, and Radu Timofte

Affiliation Alexandru Ioan Cuza University of Iasi, and ETH Zürich and University of Würzburg

Rank 7 [Greetwin](#) got a score of [5749](#) in [7265](#) seconds.  [19]

Authors Jippe Hoogeveen

Affiliation Utrecht University

Rank 8 [METATWW](#) got a score of [4265](#) in [27756](#) seconds.  [25]

Authors William Bille Meyling

Affiliation University of Copenhagen

Rank 9 [twin width fmi](#) got a score of [3231](#) in [16081](#) seconds.  [31]

Authors Lucian Trepteanu, Adrian Miclaus, Alexandru Enache, and Alexandru Popa

Affiliation Faculty of Mathematics and Computer Science, University of Bucharest

Rank 10 [jbalasin](#) got a score of 342 in 2345 seconds. [4]

Authors Jonathan Balasingham, Sebastian Wild and Viktor Zamaraev

Affiliation University of Liverpool

4.4 Strategies Used in the Heuristic Track

In the following, we give a short overview on the ideas used in the top three solvers. We refer the interested readers to the corresponding papers in the proceedings of IPEC 2023 for more details.

Winning Team. The winning solver, [GUTHM](#), uses three different strategies that are applied to the connected components of the graph separately. In the first strategy, a very fast heuristic is used to obtain an initial contraction sequence. This strategy greedily selects a vertex u of minimal red degree and then identifies the best partner v according to a scoring method. The scoring method aims to minimize the difference between the red degrees of vertices x , for which a new red edge will appear when contracting u and v , and the red degrees of vertices y , for which a red edge will be removed. The efficiency of this approach stems from the observation that few applications of a two-level breadth-first search can compute this scoring function. Then, depending on the expected twinwidth of the components, one of two possible strategies is applied to improve this first solution. If the twinwidth is sufficiently low, the authors use a sweeping approach where all contractions that do not exceed the width of the contraction sequence beyond a given threshold are performed. This threshold is iteratively adapted by random sampling. If the twinwidth of the connected component is expected to be high, the authors either contract two vertices chosen by the above heuristic (i. e., contracting a vertex of minimal red degree) or apply MinHashing to identify near-twins of large red degree. To speed up their computation, the authors use a data structure that manages the deleted nodes via the union-find data structure.

Runner-Up. The runner-up solver, [Zygosity](#), first contracts all twins and then contracts all trees down to a single pendant vertex connected by a red edge. The remaining time is used for multiple random contractions. Here, the quality of a contraction depends on the maximal red degree in the graph (which should be minimized) and the size of the intersection between the neighborhoods of the two chosen vertices. As the red degree is the more important measure, the intersection size is only used as tiebreaker. Instead of choosing a pair of vertices randomly, the solver chooses one vertex u randomly and then performs a random walk starting from u . The random walk is determined by a biased coin that depends on the density of the given graph. If the graph is sparse, the random walk takes only one step, and otherwise it takes two. To optimize the running time, the authors only use arrays to avoid cache-faults when iterating over the neighborhood of a vertex.

Third Place. The third place solver, [Red Alert](#), uses a randomized approach to generate about 10,000 possible vertex pairs for the contraction from which a certain number is chosen later. The number of chosen pairs directly depends on the remaining time: The less available, the rougher the subroutine to find the pairs. If t denotes the time used in the last iteration and T the remaining time, then about $n' \cdot t/T$ pairs are chosen if the current graph contains n' vertices. To select the vertices, the authors use different approaches based on the density of the remaining graph. If the remaining graph is dense, the pair is chosen uniformly at random. In the sparse case, a first vertex is chosen uniformly at random and then one or two of its

neighbors get sampled. To evaluate a pair, the solver uses a scoring function consisting of three parts: The most crucial part measures whether the contraction decreases the maximum red degree; the second part determines whether the maximum red degree is close to the red degree of the contracted vertex; and the third part counts the total number of red edges. If the time gets short, the authors partition the vertices of the remaining graph into equally sized buckets, which are then contracted.

4.5 Theory Award

To bridge the gap between theory and practice, we gave out a theory award to encourage submissions containing new theoretical insights. The theory award has been granted to the solver [Hydra Prime](#) by Yosuke Mizutani, David Dursteler, and Blair D. Sullivan (highlighted with a “🏆” in Section 4.1). The award was primarily given due to the development of two novel ideas: the timeline encoding and hydra decompositions. The *timeline encoding* is an innovative data structure that enables the efficient computation of the width of a contraction sequence, resulting in enhanced local search capabilities. *Hydra decompositions*, conversely, are a divide-and-conquer strategy that features compact vertex separators.

5 PACE Organization

The program committee of PACE 2023 consisted of the co-chairs Max Bannach and Sebastian Berndt. During the competition, the members of the steering committee were:

- (since 2016) Holger Dell (Goethe University Frankfurt and IT University of Copenhagen)
- (since 2019) Johannes Fichte (Linköping University)
- (since 2019) Markus Hecher (MIT)
- (since 2016) Bart M. P. Jansen (chair) (Eindhoven University of Technology)
- (since 2020) Łukasz Kowalik (University of Warsaw)
- (since 2021) André Nichterlein (Technical University of Berlin)
- (since 2020) Marcin Pilipczuk (University of Warsaw)
- (since 2022) Christian Schulz (Heidelberg University)
- (since 2020) Manuel Sorge (Technische Universität Wien)

6 Conclusion 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 2023 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.

PACE 2024. We look forward to the next edition, which will focus on ONE-SIDED CROSSING MINIMIZATION and will be chaired by Philipp Kindermann. Detailed information will be posted on the website of the competition at pacechallenge.org.

References

- 1 Barla Adam, Blažej Václav, Červený Radovan, Dvořák Michal, Knop Dušan, Koleda Jozef, Pokorný Jan, Šťastný Petr, and Suchý Ondřej. G2oat solver for pace 2023 (twinwidth) exact track, June 2023. doi:10.5281/zenodo.7997817.
- 2 Sachin Agarwal, Aman Jain, Talika Gupta, and Srinibas Swain. Hatter: Hybrid approach to twinwidth by taming red, June 2023. doi:10.5281/zenodo.8045969.
- 3 Andrei Arhire and Matei Chiriac. Uaic_twin_width: An exact twin-width algorithm, June 2023. doi:10.5281/zenodo.8010483.
- 4 Jonathan Balasingham, Sebastian Wild, and Viktor Zamaraev. Pace-2023, May 2023. doi:10.5281/zenodo.7996417.
- 5 Tomas Balyo, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors. *Proceedings of SAT Competition 2023: Solver, Benchmark and Proof Checker Descriptions*. Department of Computer Science Series of Publications B. Department of Computer Science, University of Helsinki, Finland, 2023.
- 6 Gabriel Bathie, Jérôme Boillot, Nicolas Bousquet, and Théo Pierron. Pace 2023 - tinywidth solver, May 2023. doi:10.5281/zenodo.7991737.
- 7 Pierre Bergé, Édouard Bonnet, and Hugues Déprés. Deciding twin-width at most 4 is NP-complete. In *ICALP*, volume 229 of *LIPICs*, pages 18:1–18:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- 8 Pierre Bergé, Édouard Bonnet, Hugues Déprés, and Rémi Watrigant. Approximating highly inapproximable problems on graphs of bounded twin-width. In *STACS*, volume 254 of *LIPICs*, pages 10:1–10:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.
- 9 Gaétan Berthe, Yoann Coudert–Osmont, Alexander Dobler, Laure Morelle, Amadeus Reinald, and Mathis Rocton. Doblalex/touiouidh: v1.0, June 2023. doi:10.5281/zenodo.8027196.
- 10 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphane Thomassé, and Rémi Watrigant. Twin-width III: max independent set, min dominating set, and coloring. In *ICALP*, volume 198 of *LIPICs*, pages 35:1–35:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- 11 Édouard Bonnet, Eun Jung Kim, Stéphane Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In *FOCS*, pages 601–612. IEEE, 2020.
- 12 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.
- 13 Holger Dell, Anselm Haak, Frank Kammer, Alexander Leonhardt, Johannes Meintrup, Meyer Ulrich, and Manuel Penschuck. GUTHM and GUTHMi: Exact and heuristic twin-width solvers, June 2023. doi:10.5281/zenodo.7996074.
- 14 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.
- 15 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.
- 16 David Dursteler and Yosuke Mizutani. Pace 2023 - exact, June 2023. doi:10.5281/zenodo.7996823.
- 17 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.
- 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.

- 19 Jippe Hoogeveen. Greetwin pace 2023, June 2023. doi:10.5281/zenodo.8034100.
- 20 Yinon Horev, Shiraz Shay, Sarel Cohen, Tobias Friedrich, Davis Issac, Lior Kamma, Aikaterini Niklanovits, and Kirill Simonov. Satwin, June 2023. doi:10.5281/zenodo.8047007.
- 21 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.
- 22 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.
- 23 Kenneth Langedal, Emmanuel Arrighi, Pål Grønås Drange, Farhad Vadiée, Martin Vatshelle, and Petra Wolf. Zygoty pace 2023, September 2023. doi:10.5281/zenodo.8370343.
- 24 Alex Meiburg. Timeroot/twinwidth: Pace 2023, June 2023. doi:10.5281/zenodo.8045233.
- 25 William Bille Meyling. Metatww solver for pace-2023, June 2023. doi:10.5281/zenodo.8045541.
- 26 Möller, Funk, Jakob, and Großmann. Heitwin-exact - pace challenge 2023, May 2023. doi:10.5281/zenodo.7988134.
- 27 Thomas Möller, Nikita-Nick Funk, Dennis Jakob, and Ernestine Großmann. Heitwin-heuristic - pace challenge 2023, May 2023. doi:10.5281/zenodo.7986572.
- 28 Networks project, 2017. URL: <https://www.thenetworkcenter.nl>.
- 29 André Schidler and Stefan Szeider. A SAT approach to twin-width. In *ALLENEX*, pages 67–77. SIAM, 2022.
- 30 Andre Schidler and Stefan Szeider. Computing twin-width with branch & bound - PACE Submission, June 2023. doi:10.5281/zenodo.8033459.
- 31 Lucian Trepteanu. luciantrepteanu/pace2023: pace-2023, May 2023. doi:10.5281/zenodo.7991297.
- 32 Kaio Vieira, alanctprado, emanuel juliano morais silva, and Laila Melo Vaz Lopes. emanueljuliano/pace2023: pace-2023, June 2023. doi:10.5281/zenodo.8045380.
- 33 Christopher Weyand and Marcus Wilhelm. soap1, May 2023. doi:10.5281/zenodo.7989779.
- 34 Bonnet Édouard and Duron Julien. Redalert, June 2023. doi:10.5281/zenodo.8079499.