

The PACE 2018 Parameterized Algorithms and Computational Experiments Challenge: The Third Iteration

Édouard Bonnet

Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France
edouard.bonnet@dauphine.fr

Florian Sikora

Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016, Paris, France
florian.sikora@dauphine.fr

Abstract

The Program Committee of the Third Parameterized Algorithms and Computational Experiments challenge (PACE 2018) reports on the third iteration of the PACE challenge. This year, all three tracks were dedicated to solve the STEINER TREE problem, in which, given an edge-weighted graph and a subset of its vertices called *terminals*, one has to find a minimum-weight subgraph which spans all the terminals. In Track A, the number of terminals was limited. In Track B, a tree-decomposition of the graph was provided in the input, and the treewidth was limited. Finally, Track C welcomed heuristics. Over 80 participants on 40 teams from 16 countries submitted their implementations to the competition.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Steiner tree problem, contest, implementation challenge, FPT

Digital Object Identifier 10.4230/LIPIcs.IPEC.2018.26

Funding Partially supported by the LABEX MILYON (ANR-10- LABX-0070) of Université de Lyon, within the program “Investissements d’Avenir” (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR) and project “ESIGMA” (ANR-17-CE23-0010).

Acknowledgements The PACE challenge was supported by Networks [34], an NWO Gravitation project of the University of Amsterdam, Eindhoven University of Technology, Leiden University, and the Center for Mathematics and Computer Science (CWI), by `data-experts.de`, and by *Freunde der Saarbrücker Informatik (FdSI)*. The prize money (3750 €) and travel grants (1000 € for Daniel Rehfeldt who presented SCIP-Jack during the award ceremony) were given through the generosity of Networks [34] and `data-experts.de`. We are grateful to Szymon Wasik and Jan Badura for the fruitful collaboration and for hosting the competition at `optil.io`.

1 Introduction

The Parameterized Algorithms and Computational Experiments Challenge (PACE) was conceived in Fall 2015 to deepen the relationship between parameterized algorithms and practice. It aims to:

1. Bridge the divide between the theory of algorithm design and analysis, and the practice of algorithm engineering.
2. Inspire new theoretical developments.



© É. Bonnet, F. Sikora;

licensed under Creative Commons License CC-BY

13th International Symposium on Parameterized and Exact Computation (IPEC 2018).

Editors: Christophe Paul and Michal Pilipczuk; Article No. 26; pp. 26:1–26:15

Leibniz International Proceedings in Informatics



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

3. Investigate in how far theoretical algorithms from parameterized complexity and related fields are competitive in practice.
4. Produce universally accessible libraries of implementations and repositories of benchmark instances.
5. Encourage the dissemination of these findings in scientific papers.

The first iteration of PACE was held at IPEC 2016 [13]. Since then, PACE has been mentioned as an inspiration in many papers [1, 3, 21, 22, 25, 30, 32, 40, 41]. Hisao Tamaki's companion theory paper to his winning implementations of PACE 2017 [14] won the best paper award at ESA 2017 [39]. Very recent works acknowledging PACE or inspired by the challenge include the following papers [5, 7, 31, 20, 4, 17]. At this year's ESA, it was shown that parallelized dynamic programming (DP) on tree decompositions is a competitive approach to SAT solving [23]. According to the paper, this was made possible by the excellent programs developed for the treewidth tracks of PACE 2016 and 2017. At the same conference, Bannach and Brendt provided an efficient and simple interface to DP on tree decompositions [2].

In this article, we report on the third iteration of PACE. The PACE 2018 challenge was announced on November 14, 2017. The final version of the submissions was due on May 12, 2018. We informed the participants of the result on May 14, and announced them to the public on August 22nd, during the award ceremony at the International Symposium on Parameterized and Exact Computation (IPEC 2018) in Helsinki.

2 The Steiner Tree Problem: Theory and Practice

There are many different variants of the Steiner Tree problem. They all involve the task to interconnect objects called *terminals* by using minimum-length *wires*. Here, we focus on the main variant in graphs. For a subset of edges F of an undirected graph, let $V(F)$ denote the set of vertices that are endpoints of those edges. A formal definition of the STEINER TREE problem can be stated as follows.

Input: An undirected graph G with a non-negative weight function on its edges $w : E(G) \rightarrow \mathbb{R}^+$, and a set of *terminals* $T \subseteq V(G)$.

Output: A subset of edges $F \subseteq E(G)$ minimizing $\sum_{e \in F} w(e)$ such that: $T \subseteq V(F)$ holds and $(V(F), F)$ is a connected graph.

We will interchangeably regard F and $(V(F), F)$ as the solution. In words, given an edge-weighted graph and a subset of its vertices (the *terminals*) the task is to find a minimum-weight subgraph that spans all terminals. As the weights are non-negative, any edge of a cycle in a feasible solution can be removed to obtain a solution that is still feasible but has a smaller or equal total weight. Hence, there is always a tree among the optimum solutions, which explains the second word, *Tree*, in the problem name. The first word, *Steiner*, refers to the 19th-century geometer Jakob Steiner.

We chose STEINER TREE to be the problem of PACE 2018 since it has a wide spectrum of real-life applications, including the design of VLSI, optical or wireless communication systems, and transportation networks [29]. Furthermore, as a result of the 11th DIMACS implementation challenge, there are established benchmarks for STEINER TREE and its variants. Finally, as we will see in the next section, this problem has interesting FPT algorithms.

2.1 Parameterized algorithms

In what follows, we denote the number of edges of an input graph G by n , its number of edges by m , its number of terminals by t , and its treewidth by w . *Steiner vertices* are the non-terminal vertices touched by a solution $F \subseteq E(G)$, and we denote their number by s , that is, we have $s = |V(F) \setminus T|$. As we will see, there are FPT algorithms for STEINER TREE with t as a parameter and with w as a parameter, that is, algorithms running in time $f(t)n^{O(1)}$ and $g(w)n^{O(1)}$ for some computable functions f and g .

The Dreyfus-Wagner algorithm

There is a classical algorithm from the 70s designed by Dreyfus and Wagner [16] (henceforth DW) with running time $O(3^t n + 2^t n^2 + n(n \log n + m))$. This was later improved to $O(3^t n + 2^t(n \log n + m))$ by Erickson, Monma, and Veinott [19] (henceforth EMV). It works by dynamic programming on pairs of disjoint subsets of the set of terminals T . To keep it simple, in a solution $(V(F), F)$, there is a vertex with degree at least 3 whose removal would disconnect two disjoint sets of terminals $T_1, T_2 \subset T$. One can store a lightest way of connecting T_1 and T_2 for all the pairs T_1, T_2 in a bottom-up manner. The number of pairs of disjoint sets in a universe of t elements is 3^t , which explains part of the running time of DM and EMV.

One can wonder if the exponential basis of 3 can be decreased. There has been quite a bit of work done in that direction, leading to a series of improvements going from 3 to 2. However, no submission to this year's PACE challenge have made use of these improvements, for good reason: They tend to come with a nasty blow-up in the polynomial factors. For instance, there are $O(2.5^t n^{15})$ and $O(2.1^t n^{58})$ time algorithms [24]. If the weights are bounded by some constant (and in the variant where the vertices, not the edges, are weighted), there is an algorithm in time roughly $O(2^t n^2 + nm)$, based on the so-called fast subset convolution [6].

Treewidth-based algorithms

Given a tree-decomposition $(\mathcal{T}, \{V_u\}_{u \in V(\mathcal{T})})$ of width w of the input graph G (with \mathcal{T} , a rooted tree and $\{V_u\}$, a family of subsets of $V(G)$ indexed by the nodes of \mathcal{T}), one can solve STEINER TREE in time $O^*(w^w)$ by bottom-up dynamic programming on the nodes of \mathcal{T} . Indeed, once the subtree of \mathcal{T} at $u \in V(\mathcal{T})$ is processed, the information to remember, besides the invested budget, is the subset of vertices of V_u touched by a partial solution and how these vertices are connected in the partial solution. This information can be represented by a partition of a subset of V_u . The number of such objects is bounded by $O(w^w)$ since $|V_u| \leq w + 1$, hence the size of the DP table and the running time.

For some time it was an open question if STEINER TREE and similar problems with a connectivity constraint can be solved in time $2^{O(w)}n^{O(1)}$ instead of $2^{O(w \log w)}n^{O(1)}$. This was finally resolved in 2011 with the Cut&Count technique of Cygan et al. [12]. In a nutshell, the Cut&Count technique starts by addressing the parity version of the problem: how many sets of ℓ connected edges span T , modulo 2? For this, one counts the number p of pairs (D, \mathcal{C}) where D is a set of at most ℓ edges spanning T (and D is not required to be connected) and \mathcal{C} is a cut such that every connected component of D is fully contained in one side of \mathcal{C} . If we restrict a designated terminal to always be on the *left* side of \mathcal{C} , the number of possible \mathcal{C} for a given D is $2^{\text{cc}(D)-1}$, where $\text{cc}(D)$ is the number of connected components of D . This number is 1 if D is connected and even otherwise. So, the parity of p is the answer to our question. Since computing p does not exhibit a (*global*) connectivity constraint, it can be done in time $2^{O(w)}n^{O(1)}$ by standard dynamic programming. Finally, one can distinguish the

case *zero solutions* from the case *non-zero even number of solutions* by using the Isolation Lemma. This lemma implies that by giving uniformly random integer weights between 1 and $100\ell m$ to edges, there will be, with high probability, exactly one solution of minimum total weight (provided, of course, that there was a solution to begin with). For this total weight, we will therefore find an odd number of solutions, and may confidently report it.

One might then ask for a deterministic $2^{O(w)}n^{O(1)}$ time algorithm, which could handle weighted instances. The rank-based approach of Bodlaender et al. [8] does exactly that. The main idea is that when performing the standard $O^*(w^w)$ DP, one does not need to keep *every* partition of a bag. Instead, one can prune the partitions in the following way. If \mathcal{S} is the set of partitions of a bag achievable by partial solutions of a certain cost, then one can safely restrict to working with a subset $\mathcal{S}' \subseteq \mathcal{S}$, such that: if there is a $\Pi_1 \in \mathcal{S}$ compatible with a partition Π^* generated by a partial solution in the rest of the graph, then there should be at least one $\Pi_2 \in \mathcal{S}'$ also compatible with Π^* . By *compatible*, we mean that the two partitions would fully connect the common subset of the bag touched by the partial solutions. If one represents compatibility by a square matrix in \mathbb{F}_2 whose rows and columns are indexed by the partitions (with a 1 in the entries corresponding to compatible partitions, and a 0 otherwise), then it can be observed that any single row/column of a linearly dependent family of rows/columns can be safely removed. Indeed, for every index of the deleted vector containing a 1, there has to be another vector of the family also containing a 1 at this position. Then it is proved that the rank of the matrix with all the partitions is $2^{O(w)}$, and that Gaussian elimination can be done in the same time without having to explicitly store or compute all $O^*(w^w)$ permutations. This three-act story is told with the exact balance of details and conciseness in the book on parameterized complexity by Cygan et al. [11].

Known theoretical obstacles

The STEINER TREE problem parameterized by the number s of Steiner vertices is $W[2]$ -hard [33]. Furthermore it is unlikely that there is an algorithm running in time $n^{s-\epsilon}$. Such an algorithm would indeed give, by a simple reduction, an algorithm in $n^{k-\epsilon}$ for k -DOMINATING SET, contradicting the Strong Exponential Time Hypothesis [38]. Theory also says that one cannot expect very efficient preprocessing: there is no polynomial kernel in $s + t$ (the total number of vertices in $V(F)$) unless the polynomial-time hierarchy collapses [15]. Although, in practice, there are dozens of listed reductions rules which can sometimes completely solve large real-world instances.

Approximations

We hastily touch the approximability status of STEINER TREE as it might be relevant to heuristics. A minimum spanning tree in the subgraph induced by the terminals of the metric closure yields an approximation ratio of 2 [27]. There is a 1.39-approximation based on a scheme called *iterative randomized rounding* of an LP-formulation [9], and a combinatorial 1.55-approximation [37]. However, STEINER TREE is APX-hard: there is no 1.01-approximation unless $P=NP$ [10].

2.2 DIMACS challenge

In 2014, the 11th (and latest) DIMACS implementation challenge was also dedicated to solving Steiner Tree problems, and extensive libraries of instances existed prior to our challenge. As we will detail later, we mainly selected our instances from the Steinlib sets.

Furthermore, there were already very efficient programs solving Steiner Tree and its variants. One successful program during the DIMACS competition, SCIP-Jack, also participated in our challenge. SCIP-Jack is a branch-and-cut algorithm based on the free MIP-solver SCIP. As such, it provides a natural baseline and comparison for the FPT approaches.

3 Selection of the instances and rules

Before we selected the instances, we first defined the precise goals of each track. In light of the efficient parameterized algorithms described in the previous paragraphs and the objectives of PACE, it will not come as a surprise that we decided that Track A would have instances where the number t of terminals is relatively small, while in Track B the treewidth w of the graphs would be relatively small.

Generating artificial instances for which a careful optimization of the Dreyfus-Wagner algorithm (for Track A) or of the rank-based approach for dynamic programming on tree decompositions (for Track B) would get a decisive advantage compared to a generic solver (ILP, SAT) was not our goal. So, we selected our instances among established benchmarks (mainly Steinlib, PUC, GAP, Vienna), which were also used four years ago in the DIMACS challenge on Steiner Tree.

3.1 Track A

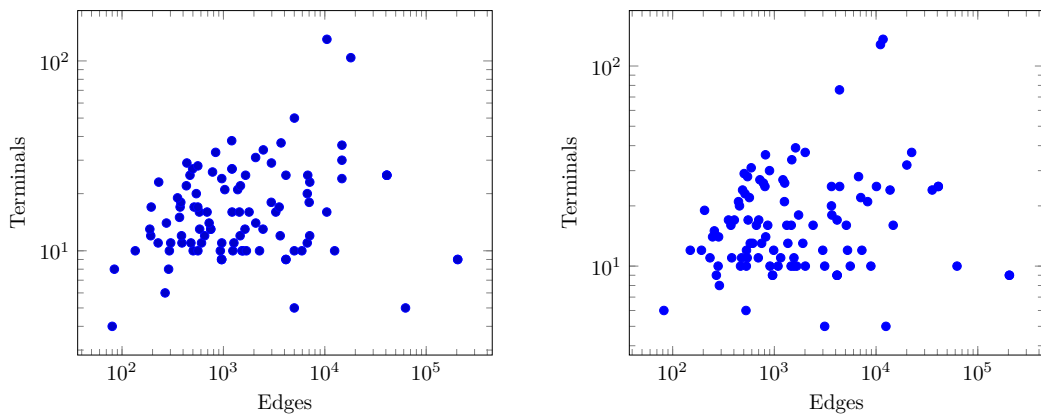
For this track, we picked quite a lot of instances with a VLSI application. They are grid graphs with rectangular holes and Manhattan distance weights (in the ALUE, ALUT, DIW, DMXA, GAP, MSM, TAQ, and LIN sets). We used randomly generated sparse instances meant to be resistant to preprocessing (in the E, I160, I640, PUC sets). We also included industrial instances stemming from wire routing problems (in the WRP3 and WRP4 sets), and real geographical networks in the form of complete graphs with Euclidean distances (X set).

Among these instance sets, we selected 200 instances: 100 public (given to the contestants) and 100 private (used for the evaluation), with 4 to 136 terminals. The median number of terminals is 16 and its mean is 19.4, while 1480 is the mean number of vertices, and 8515, the mean number of edges (see Figure 1). On more than 20 of those instances, top implementations of the DIMACS challenge, such as Staynerd, take tens of minutes. We sorted the instances by increasing (t, m) in the lexicographic order.

The main rule for Track A and B was that the algorithm (deterministic or randomized) should be exact. A sub-optimal output (in the public or private instances) would therefore mean disqualification. The final score is the number of private instances solved on the `optil.io` platform with 30 minutes per instance.

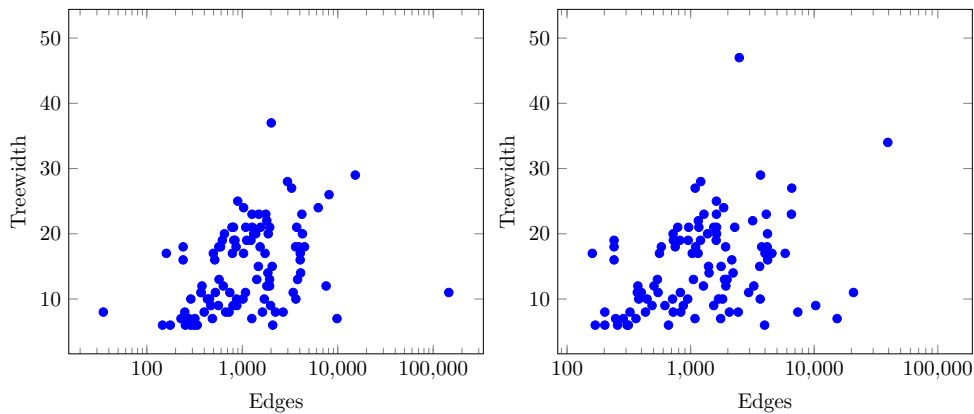
3.2 Track B

For this track, we used a lot of rectilinear instances with a low treewidth but a high number of terminals (in the ES100FST, ES500FST, ES1000FST sets, where the number between ES and FST corresponds to the number of terminals). We also used many instances from WRP3 and WRP4 with an intermediate number of terminals (≈ 50). A tree decomposition of almost always minimum width was given with the input. We computed these decompositions using the winning implementations of the treewidth track of PACE 2017 by Strasser and by Tamaki and his team [14].



■ **Figure 1** The distribution ($\#edges, \#terminals$) of the public (left) and private (right) instances of Track A in log-log plot.

On the 200 instances that we picked, the average number of vertices is 1490, the average number of edges is 2847, the average number of terminals is 606, and its median is 100. Furthermore the width of the given tree decompositions range from 6 to 47 with a median of 19.5 (see Figure 2). We sorted the instances by increasing (w, m) . The rules were the same as in Track A.



■ **Figure 2** The distribution ($\#edges, treewidth$) of the public (left) and private (right) instances of Track B in lin-log plot.

3.3 Track C

Finally in Track C, dedicated to heuristics, we chose large and difficult instances with many terminals and high treewidth. To this end, we used the hardest instances of Steinlib. We also added a lot of instances from the Vienna set. These instances were generated from real-world telecommunication networks by Ivana Ljubic's group at the University of Vienna. A majority of the selected instances cannot be solved within one hour by the state-of-the-art program. In several cases, the actual optimum is unknown.

The average number of vertices is 27K, the average number of edges is 48K, and the average number of terminals is 1114, with a median at 360.5 (see Figure 3). Finally the best

treewidth upper bound (obtained with the heuristics of previous PACE) on these instances is almost always above 40. We sorted the instances by increasing (t, m) .

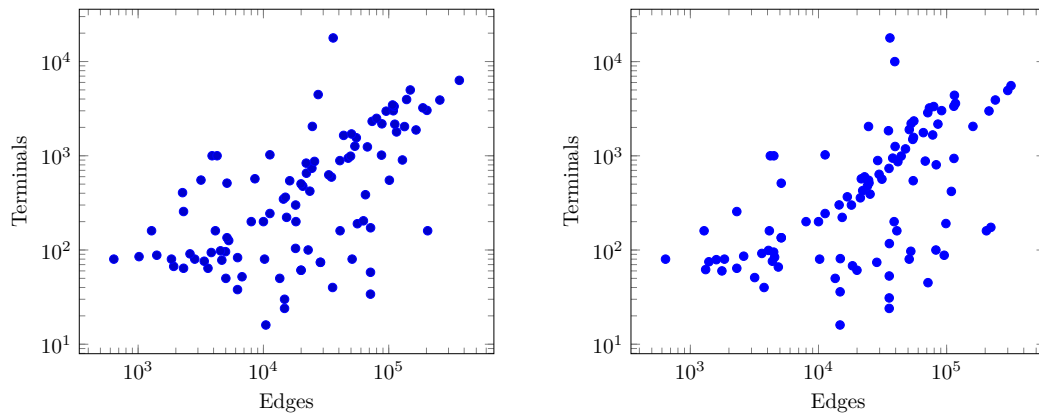


Figure 3 The distribution ($\#edges, \#terminals$) of the public (left) and private (right) instances of Track C in log-log plot.

In track C, the final score to maximize is the sum over all the private instances of the ratio opt/sol where opt is the optimum (if known) or the best known upper bound (otherwise) and sol is the value of the proposed output. Outputting a non-feasible solution would not disqualify the submission, but would give a score of 0 on that particular instance.

4 Participation and results

This year, we had over 40 teams and 80 participants coming from 16 countries and four continents: including Austria, Brazil, Canada, Czechia, Denmark, England, Finland, France, Germany, India, Japan, Mexico, the Netherlands, Norway, Poland, and Romania.

The number of teams and participants both doubled compared to PACE 2017. To be precise, the above numbers correspond to teams and participants who sent a final implementation to at least one track. If we also count people who uploaded some code on the `optil.io` platform but dropped out of the competition, the number of teams exceeds 50 and the number of participants exceeds 100. If we count the teams with their multiplicity (that is, the number of tracks in which they participated), this number exceeds 75.

4.1 Track A

Yoichi Iwata and Takuto Shigemura won this track by solving 95 private instances. Their algorithm builds upon the dynamic programming of Erickson-Monma-Veinott (EMV). A technical lemma permitted them to prune a lot of entries in the DP table. This happens to also solve instances with small treewidth even when the number of terminals is relatively large.

Krzysztof Maziarz and Adam Polak got the second place by preprocessing and improving Dreyfus-Wagner (DW) with the heuristic ideas presented by Hougardy et al. [28]. The latter consists of running a shortest path algorithm (Dijkstra, A^* , etc.) in the partial solutions in order to prune the search space.

Koch and Rehfeldt took the third place with their program SCIP-Jack which has three main components. First, known and new reduction methods simplify the instance. Secondly, a set of heuristics provides upper and lower bounds. Finally, a reduction (of many variants)

Country	Teams	Participants
Austria	2	4
Brazil	1	3
Canada	1	1
Czechia	2	4
Denmark	1	1
England	1	1
Finland	1	1
France	4	7
Germany	4	5
India	6	12
Japan	4	8
Mexico	1	4
Netherlands	2	6
Norway	2	4
Poland	2	11
Romania	1	3
Total	35	75

■ **Table 1** Participation per country (based on the initial registration form; more teams and participants uploaded their code on `optil.io` afterwards).

to the directed Steiner tree problem is performed and the core of the algorithm is a branch-and-cut approach using the MIP-solver SCIP (developed by a superset of this team).

At the fourth place, albeit still very close to the first place, solving 92 instances, Andre Schidler, Johannes Fichte, and Markus Hecher used reduction rules presented by Rehfeldt [36], including the so-called *dual ascent* and the improved DW by Hougardy et al. [28].

All the other teams implemented some refinements of DW or EMV.

- **1st place, 450 €:** Yoichi Iwata and Takuto Shigemura (team `wata&sigma` from Japanese National Institute of Informatics and University of Tokyo) solved 95 out of 100 instances github.com/wata-orz/steiner_tree
- **2nd place, 350 €:** Krzysztof Maziarz and Adam Polak (team Jagiellonian from Jagiellonian University) solved 94 out of 100 instances <https://bitbucket.org/kris maz/pace2018>
- **3rd place, 300 €:** Thorsten Koch and Daniel Rehfeldt (team `reko` from Zuse Institute Berlin and TU Berlin) solved 93 out of 100 instances github.com/dRehfeldt/scipjack/
- **4th place, 225 €:** Andre Schidler, Johannes Fichte, and Markus Hecher (team `TUW` from TU Vienna) solved 92 out of 100 instances github.com/ASchidler/pace17/
- **5th place:** Krzysztof Kiljan, Dominik Klemba, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Mateusz Radecki, and Michał Ziobro (team `UWarsaw` from University of Warsaw and Jagiellonian University) solved 67 out of 100 instances https://bitbucket.org/marcin_pilipczuk/pace2018-steiner-tree
- **6th place:** Suhas Thejaswi (team `Noname` from Aalto University) solved 66 out of 100 instances github.com/suhastheju/pace-2018-exact

- **6th place:** Peter Mitura and Ondřej Suchý (team FIT CTU in Prague from the Czech Technical University) solved 66 out of 100 instances
github.com/PMitura/pace2018
- **6th place:** Johannes Varga (team johannes from TU Vienna) solved 66 out of 100 instances
github.com/josshy/st-tree
- **9th place:** Saket Saurabh, P. S. Srinivasan, and Prafullkumar Tale (team SSPSPT from the Institute Of Mathematical Sciences, HBNI, Chennai and the International Institute of Information Technology, Bangalore) solved 48 out of 100 instances
github.com/pptale/PACE18

One can see two well-defined clusters (90-95 and 65-70). The following implementation would have led the second cluster, if not for a small bug that showed only on one private instance.

- **Honorable mention:** Sharat Ibrahimpur (team the65thbit from University of Waterloo) solved 69 out of 100 instances but was incorrect on 1 instance
github.com/sharat1105/PACE2018
- **11th place:** S. Vaishali and Rathna Subramanian (team PCCoders from PSG College of Technology, Coimbatore) solved 14 out of 100 instances but was incorrect on several instances
github.com/ammuv/PACE-2018-
- **12th place:** R. Vijayaragunathan, N. S. Narayanaswamy, and Rajesh Pandian M. (team Resilience from TCS Lab, Indian Institute of Technology, Madras) solved 9 instances but was incorrect on several instances
github.com/mrprajesh/pace2018

4.2 Track B

In this track with low treewidth, the developers of SCIP-Jack, Thorsten Koch and Daniel Rehfeldt, took the first place by solving 92 of the 100 instances in the private set. They do not directly exploit the tree-decomposition given in input, but it is likely that the low treewidth indirectly translates into higher efficiency of their preprocessing, heuristics, and/or branch-and-cut. Otherwise, it is hard to explain how the same program, discarding the tree-decomposition, won Track B by solving 15 more instances than the winner of Track A (where they finished third).

At the second place, Yoichi Iwata and Takuto Shigemura implemented the standard $O^*(w^w)$ dynamic programming on the tree-decomposition. When the treewidth becomes too large, they switch to their improved EMV algorithm. They observe that their algorithm for Track A performs well when the treewidth is low since the existence of small separators prunes a lot of entries in the DP table.

Completing the podium, Tom van der Zanden added to the rank-based approach a handful of nice observations, crediting some of them to Luuk van der Graaff's master thesis. This results in the only program of the competition solving all the instances with treewidth at most 15. In particular, this implementation is the only one to solve instance 39 of the private set, which has 5K+ vertices, 20K+ edges, 2.4K+ terminals, and treewidth 11.

- **1st place, 450 €:** Thorsten Koch and Daniel Rehfeldt (team reko from Zuse Institute Berlin and TU Berlin) solved 92 out of 100 instances
github.com/dRehfeldt/scipjack/

- **2nd place, 350 €:** Yoichi Iwata and Takuto Shigemura (team wata&sigma from the Japanese National Institute of Informatics and the University of Tokyo) solved 77 out of 100 instances
github.com/wata-orz/steiner_tree
- **3rd place, 300 €:** Tom van der Zanden (team Tom from Utrecht University) solved 58 out of 100 instances
github.com/TomvdZanden/SteinerTreeTW
- **4th place:** Peter Mitura and Ondřej Suchý (team FIT CTU in Prague from the Czech Technical University) solved 52 out of 100 instances
github.com/PMitura/pace2018
- **4th place:** Yasuaki Kobayashi (team Yasu from Kyoto University) solved 52 out of 100 instances
https://bitbucket.org/yasu0207/steiner_tree
- **6th place:** Akio Fujiyoshi (team CBGfinder from Ibaraki University) solved 49 out of 100 instances
github.com/akio-fujiyoshi/CBGfinder_for_steiner_tree_problem
- **7th place:** Krzysztof Kiljan, Dominik Klemba, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Mateusz Radecki, and Michał Ziobro (team UWarsaw from University of Warsaw and Jagiellonian University) solved 33 out of 100 instances
https://bitbucket.org/marcin_pilipczuk/pace2018-steiner-tree
- **7th place:** Dilson Guimarães, Guilherme Gomes, João Gonçalves, and Vinícius dos Santos (team lapo from LAPO-UFMG) solved 33 out of 100 instances
github.com/dilsonguim/pace2018

The rest of the teams implemented the rank-based approach of Bodlaender et al. [8] as the main or secondary component. At the shared fourth place, Peter Mitura and Ondřej Suchý (FIT CTU in Prague)’s implementation is the fastest of the competition on average, while Yasuaki Kobayashi’s program also switches to a terminal-based approach when the treewidth becomes too large. Table 2 summarizes the approach used by all the teams.

Team	score	treewidth approach	first unsolved	switching to DW-like	criterion to switch
reko	92	no	26, $w = 9$	no	–
wata_sigma	77	w^w DP	26, $w = 9$	EMV	$w > 10$ or ($w > 8$ and $t < 300$)
Tom	58	$2^{O(w)}$ rank-based	54 , $w = 16$	EMV	$3^t < 5^w$
FIT CTU in Prague	52	$2^{O(w)}$ rank-based	39, $w = 11$	no	–
yasu	52	$2^{O(w)}$ rank-based	13, $w = 7$	DW	$w > 15$
fujiyoshi	49	$2^{O(w)}$ rank-based	39, $w = 11$	no	–
UWarsaw	33	$2^{O(w)}$ rank-based	32, $w = 10$	no	–
lapo	33	$2^{O(w)}$ rank-based	13, $w = 7$	no	–

■ **Table 2** For each team: *score* is the number of private instances solved, *treewidth approach*, the treewidth-based algorithm used, if any, *first unsolved*, the number of the first unsolved private instance (we recall that the instances of Track B were sorted lexicographically by increasing (w, e)) and the corresponding value of the treewidth, *switching to DW-like*, if the treewidth approach is substituted to a terminal-based algorithm (DW = Dreyfus-Wagner algorithm, EMV = Erickson-Monma-Veinott algorithm), and *criterion to switch* what is the test to switch to such an algorithm.

Since the instances were edge-weighted, no submission tried to use the Cut&Count technique. It would be interesting to compare the speed of the rank-based and the Cut&Count techniques on unweighted instances of relatively low treewidth. Another direction would be to make Cut&Count work on edge-weighted instances, in practice and/or in theory.

4.3 Track C

This track attracted the most participants. Regarding the instance sizes, it should be observed that a naive implementation (without preprocessing) of the 2-approximation algorithm does not finish within the time limit. The contributions use either meta-heuristics (an evolutionary algorithm for the winning team, iterated local-search with noising for the third team, simulated annealing, etc.), or start from some solution (a spanning tree, an arbitrary feasible solution, or a 2-approximation) and then improve it with local search. Radek Hušek and teammates (Team Tarken) used a parameterized approximation approach [18].

- **1st place, 450 €:** Emmanuel Romero Ruiz, Emmanuel Antonio Cuevas, Irwin Enrique Villalobos López, and Carlos Segura González (CIMAT Team from the Center for Research in Mathematics, Guanajuato) got an average ratio of 99.93/100
github.com/HeathcliffAC/SteinerTreeProblem
- **2nd place, 350 €:** Thorsten Koch and Daniel Rehfeldt (team reko from Zuse Institute Berlin and TU Berlin) got an average ratio of 99.85/100
github.com/dRehfeldt/scipjack/
- **3rd place, 300 €:** Martin Josef Geiger (team MJG from HSU Hamburg) [26] got an average ratio of 99.80/100
<https://data.mendeley.com/datasets/yf9vpkgwdr/1>
- **4th place, 225 €:** Radek Hušek, Tomáš Toufar, Dušan Knop, Tomáš Masařík, and Eduard Eiben (team CUiB from Charles University, Prague and University of Bergen) got an average ratio of 99.72/100
github.com/goderik01/PACE2018
- **5th place:** Emmanuel Arrighi and Mateus de Oliveira Oliveira (team Gardeners from ENS Cachan and University of Bergen) got an average ratio of 98.93/100
github.com/SteinerGardeners/TrackC-Version1
- **6th place:** Krzysztof Kiljan, Dominik Klemba, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Mateusz Radecki, and Michał Ziobro (team UWarsaw from University of Warsaw and Jagiellonian University) got an average ratio of 98.27/100
https://bitbucket.org/marcin_pilipczuk/pace2018-steiner-tree
- **7th place:** Stéphane Grandcolas (team SGLS from LIS Marseille) got an average ratio of 97.54/100
<http://www.dil.univ-mrs.fr/~gcolas/sgls.c>
- **8th place:** Max Hort, Marciano Geijselaers, Joshua Scheidt, Pit Schneider, and Tahmina Begum (team JMMPT from the Department of Data Science and Knowledge Engineering, Maastricht University) got an average ratio of 97.15/100
github.com/maxhort/Pacechallenge-TrackC/
- **9th place:** Dimitri Watel and Marc-Antoine Weisser (team DoubleDoubleU from SAMOVAR-ENSIIE and LRI-Centrale-Supelec) got an average ratio of 96.92/100
github.com/mouton5000/PACE2018/
- **10th place:** R. Vijayaragunathan, N. S. Narayanaswamy, and Rajesh Pandian M. (team Resilience from TCS Lab and the Indian Institute of Technology Madras) got an average ratio of 94.57/100
github.com/mrprajesh/pace2018
- **11th place:** Sharat Ibrahimpur (team the65thbit from University of Waterloo) got an average ratio of 94.37/100
github.com/sharat1105/PACE2018

Team name	Cumul. ratio	# Best	# $\leq 0.1\%$	# $\leq 0.5\%$	# $\leq 1\%$	# $\leq 2\%$	# $\leq 3\%$
CIMAT Team	99.93	34	49	95	100	100	100
reko	99.85	78	81	88	93	99	100
MJG	99.80	24	58	87	95	98	100
CUiB	99.72	19	48	82	89	97	100

■ **Table 3** For the first four teams: their final score, how many times they computed the best solution, and how many times their solution is $x\%$ close to the best solution, on the private set.

- **12th place:** Saket Saurabh, P. S. Srinivasan, and Prafullkumar Tale (team SSPSPT from the Institute Of Mathematical Sciences, HBNI, Chennai and the International Institute of Information Technology, Bangalore) got an average ratio of 82.61/100
github.com/pptale/PACE18
- **13th place:** Harumi Haraguchi, Hiroshi Arai, Shiyougo Akiyama, and Masaki Kubonoya (team Haraguchi laboratory from Ibaraki University) got an average ratio of 80.73/100
github.com/Harulabo/pace2018

Eleven more teams participated in this track but did not finalize their submission by sharing a link to their code. Table 3 shows that the second team (reko) was more often the one reporting a best solution. However, the first team (CIMAT Team) was more consistently very close to those best solutions.

5 PACE organization

In September 2018, the PACE 2018 Program Committee transferred to the Steering Committee. The Steering Committee and the PACE 2018 Program Committee are as follows.

Steering committee:	Édouard Bonnet	ENS Lyon
	Holger Dell	Saarland Informatics Campus
	Thore Husfeldt	ITU Copenhagen & Lund University
	Bart M. P. Jansen (chair)	Eindhoven University of Technology
	Petteri Kaski	Aalto University
	Christian Komusiewicz	Philipps-Universität Marburg
	Frances A. Rosamond	University of Bergen
Track A, B, C:	Florian Sikora	Paris-Dauphine University
	Édouard Bonnet	ENS Lyon
	Florian Sikora	Paris-Dauphine University

The Program Committee of PACE 2019 will be chaired by Johannes Fichte (TU Dresden) and Markus Hecher (TU Vienna).

6 Conclusion

We thank all the participants for their enthusiasm and look forward to the next PACE. We are particularly happy that this edition attracted many people outside the parameterized complexity community, and wish that this will continue for the future editions.

We welcome anyone who is interested to add their name to the mailing list on the website [35] to receive PACE updates and join the discussion. In particular, plans for PACE 2019 will be posted there.

References

- 1 Michael Abseher, Nysret Musliu, and Stefan Woltran. htd - A free, open-source framework for (customized) tree decompositions and beyond. In Domenico Salvagnin and Michele Lombardi, editors, *Proceedings of the 14th International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR '17)*, volume 10335 of *Lecture Notes in Computer Science*, pages 376–386. Springer, 2017. doi:10.1007/978-3-319-59776-8_30.
- 2 Max Bannach and Sebastian Berndt. Practical access to dynamic programming on tree decompositions. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 6:1–6:13, 2018. URL: <https://doi.org/10.4230/LIPIcs.ESA.2018.6>, doi:10.4230/LIPIcs.ESA.2018.6.
- 3 Max Bannach, Sebastian Berndt, and Thorsten Ehlers. Jdrasil: A modular library for computing tree decompositions. In Costas S. Iliopoulos, Solon P. Pissis, Simon J. Puglisi, and Rajeev Raman, editors, *Proceedings of the 16th International Symposium on Experimental Algorithms (SEA 2017)*, volume 75 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:21, Dagstuhl, Germany, 2017. URL: <http://drops.dagstuhl.de/opus/volltexte/2017/7605>, doi:10.4230/LIPIcs.SEA.2017.28.
- 4 Max Bannach, Sebastian Berndt, Thorsten Ehlers, and Dirk Nowotka. SAT-encodings of tree decompositions. *SAT COMPETITION 2018*, page 72, 2018.
- 5 Sebastian Berndt. Computing tree width: From theory to practice and back. In *Conference on Computability in Europe*, pages 81–88. Springer, 2018.
- 6 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 67–74, 2007. URL: <http://doi.acm.org/10.1145/1250790.1250801>, doi:10.1145/1250790.1250801.
- 7 Johannes Blum and Sabine Storandt. Computation and growth of road network dimensions. In *International Computing and Combinatorics Conference*, pages 230–241. Springer, 2018.
- 8 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. URL: <https://doi.org/10.1016/j.ic.2014.12.008>, doi:10.1016/j.ic.2014.12.008.
- 9 Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013. URL: <http://doi.acm.org/10.1145/2432622.2432628>, doi:10.1145/2432622.2432628.
- 10 Miroslav Chlebík and Janka Chlebíková. The Steiner tree problem on graphs: Inapproximability results. *Theor. Comput. Sci.*, 406(3):207–214, 2008. URL: <https://doi.org/10.1016/j.tcs.2008.06.046>, doi:10.1016/j.tcs.2008.06.046.
- 11 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. URL: <https://doi.org/10.1007/978-3-319-21275-3>, doi:10.1007/978-3-319-21275-3.
- 12 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159, 2011. URL: <https://doi.org/10.1109/FOCS.2011.23>, doi:10.1109/FOCS.2011.23.
- 13 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 Jiong Guo and Danny Hermelin, editors, *Proceedings of the 11th In-*

- ternational Symposium on Parameterized and Exact Computation (IPEC 2016), volume 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:9, Dagstuhl, Germany, 2017. URL: <http://drops.dagstuhl.de/opus/volltexte/2017/6931>, doi: 10.4230/LIPIcs.IPEC.2016.30.
- 14 Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration. In *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, pages 30:1–30:12, 2017. URL: <https://doi.org/10.4230/LIPIcs.IPEC.2017.30>, doi:10.4230/LIPIcs.IPEC.2017.30.
 - 15 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Trans. Algorithms*, 11(2):13:1–13:20, 2014. URL: <http://doi.acm.org/10.1145/2650261>, doi:10.1145/2650261.
 - 16 Stuart E Dreyfus and Robert A Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971.
 - 17 Eugene F Dumitrescu, Allison L Fisher, Timothy D Goodrich, Travis S Humble, Blair D Sullivan, and Andrew L Wright. Benchmarking treewidth as a practical component of tensor-network-based quantum simulation. *arXiv preprint arXiv:1807.04599*, 2018.
 - 18 Pavel Dvořák, Andreas Emil Feldmann, Dusan Knop, Tomáš Masarík, Tomas Toufar, and Pavel Veselý. Parameterized approximation schemes for Steiner trees with small number of Steiner vertices. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 26:1–26:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. URL: <https://doi.org/10.4230/LIPIcs.STACS.2018.26>, doi: 10.4230/LIPIcs.STACS.2018.26.
 - 19 Ranel E. Erickson, Clyde L. Monma, and Arthur F. Veinott Jr. Send-and-split method for minimum-concave-cost network flows. *Math. Oper. Res.*, 12(4):634–664, 1987. URL: <https://doi.org/10.1287/moor.12.4.634>, doi:10.1287/moor.12.4.634.
 - 20 Johannes K Fichte, Markus Hecher, Neha Lodha, and Stefan Szeider. An smt approach to fractional hypertree width. *technical report*, 2018.
 - 21 Johannes Klaus Fichte, Markus Hecher, Michael Morak, and Stefan Woltran. Answer set solving with bounded treewidth revisited. In Marcello Balduccini and Tomi Janhunen, editors, *Proceedings of 14th International Conference on Logic Programming and Nonmonotonic Reasoning - (LPNMR 2017)*, volume 10377 of *Lecture Notes in Computer Science*, pages 132–145. Springer, 2017. doi:10.1007/978-3-319-61660-5_13.
 - 22 Johannes Klaus Fichte, Markus Hecher, Michael Morak, and Stefan Woltran. DynASP2.5: Dynamic programming on tree decompositions in action. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2017. doi: 10.4230/LIPIcs.IPEC.2017.17.
 - 23 Johannes Klaus Fichte, Markus Hecher, Stefan Woltran, and Markus Zisser. Weighted model counting on the GPU by exploiting small treewidth. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 28:1–28:16, 2018. URL: <https://doi.org/10.4230/LIPIcs.ESA.2018.28>, doi:10.4230/LIPIcs.ESA.2018.28.
 - 24 Bernhard Fuchs, Walter Kern, Daniel Mölle, Stefan Richter, Peter Rossmanith, and Xinhui Wang. Dynamic programming for minimum Steiner trees. *Theory Comput. Syst.*, 41(3):493–500, 2007. URL: <https://doi.org/10.1007/s00224-007-1324-4>, doi: 10.1007/s00224-007-1324-4.
 - 25 Serge Gaspers, Joachim Gudmundsson, Mitchell Jones, Julián Mestre, and Stefan Rümmele. Turbocharging treewidth heuristics. In Jiong Guo and Danny Hermelin, editors, *Proceedings*

- of the 11th International Symposium on Parameterized and Exact Computation (IPEC 2016), volume 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:13, Dagstuhl, Germany, 2017. URL: <http://drops.dagstuhl.de/opus/volltexte/2017/6932>, doi:10.4230/LIPIcs.IPEC.2016.13.
- 26 Martin Josef Geiger. Implementation of a metaheuristic for the steiner tree problem in graphs. Mendeley Data. v1. doi:10.17632/yf9vpkgwdr.1.
 - 27 Edgar N Gilbert and Henry O Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.
 - 28 Stefan Hougardy, Jannik Silvanus, and Jens Vygen. Dijkstra meets Steiner: A fast exact goal-oriented Steiner tree algorithm. *Math. Program. Comput.*, 9(2):135–202, 2017. URL: <https://doi.org/10.1007/s12532-016-0110-1>, doi:10.1007/s12532-016-0110-1.
 - 29 Frank K Hwang, Dana S Richards, and Pawel Winter. *The Steiner tree problem*, volume 53. Elsevier, 1992.
 - 30 Yoichi Iwata. Linear-time kernelization for feedback vertex set. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.68.
 - 31 Krzysztof Kiljan and Marcin Pilipczuk. Experimental evaluation of parameterized algorithms for feedback vertex set. *arXiv preprint arXiv:1803.00925*, 2018.
 - 32 Neha Lodha, Sebastian Ordyniak, and Stefan Szeider. SAT-encodings for special treewidth and pathwidth. In *Proceedings of the 20th International Conference on Theory and Applications of Satisfiability Testing (SAT 2017)*, volume 10491 of *Lecture Notes in Computer Science*, pages 429–445. Springer, 2017. doi:10.1007/978-3-319-66263-3_27.
 - 33 Daniel Mölle, Stefan Richter, and Peter Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. *Theory Comput. Syst.*, 43(2):234–253, 2008. URL: <https://doi.org/10.1007/s00224-007-9089-3>, doi:10.1007/s00224-007-9089-3.
 - 34 Networks project, 2017. URL: <http://www.thenetworkcenter.nl>.
 - 35 Parameterized Algorithms and Computational Experiments website, 2015–2018. URL: <https://pacechallenge.org>.
 - 36 Daniel Rehfeldt. A generic approach to solving the Steiner tree problem and variants. Master’s thesis, 2015.
 - 37 Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM Journal on Discrete Mathematics*, 19(1):122–134, 2005.
 - 38 Ondrej Suchý. Extending the kernel for planar steiner tree to the number of steiner vertices. *Algorithmica*, 79(1):189–210, 2017. URL: <https://doi.org/10.1007/s00453-016-0249-1>, doi:10.1007/s00453-016-0249-1.
 - 39 Hisao Tamaki. Positive-instance driven dynamic programming for treewidth. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:13, Dagstuhl, Germany, 2017. doi:10.4230/LIPIcs.ESA.2017.68.
 - 40 Tom C. van der Zanden and Hans L. Bodlaender. Computing treewidth on the GPU. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, *Leibniz International Proceedings in Informatics (LIPIcs)*, Dagstuhl, Germany, 2017. doi:10.4230/LIPIcs.IPEC.2017.29.
 - 41 Rim van Wersch and Steven Kelk. ToTo: An open database for computation, storage and retrieval of tree decompositions. *Discrete Applied Mathematics*, 217:389–393, 2017. doi:10.1016/j.dam.2016.09.023.