The PACE 2021 Parameterized Algorithms and Computational Experiments Challenge: Cluster Editing

Leon Kellerhals
Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Germany

Tomohiro Koana
Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Germany

André Nichterlein
Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Germany

Philipp Zschoche
Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Germany

Abstract

The Parameterized Algorithms and Computational Experiments challenge (PACE) 2021 was devoted to engineer algorithms solving the NP-hard Cluster Editing problem, also known as Correlation Clustering: Given an undirected graph the task is to compute a minimum number of edges to insert or remove in a way that the resulting graph is a cluster graph, that is, a graph in which each connected component is a clique.

Altogether 67 participants from 21 teams, 11 countries, and 3 continents submitted their implementations to the competition. In this report, we describe the setup of the challenge, the selection of benchmark instances, and the ranking of the participating teams. We also briefly discuss the approaches used in the submitted solvers.

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

Keywords and phrases Correlation Clustering, Cluster Editing, Algorithm Engineering, FPT, Kernelization, Heuristics

Category Invited Paper

Funding This work has been partially supported by the DFG project FPTinP (NI 369/18). The publication of the proceedings of PACE 2021 has been supported by the Algorithmics and Computational Complexity group at Technische Universität Berlin.
Tomohiro Koana: Supported by the DFG project DiPa (NI 369/21)

Acknowledgements The PACE challenge was supported by Networks [1] and Technische Universität Berlin. The prize money (€4000) was generously provided by the *Networks* [1], 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 thank Aleksander Figiel (Technische Universität Berlin) for supporting us with scripts in the data collection process.

1 Introduction

The Parameterized Algorithms and Computational Experiments Challenge (PACE) is an annually held algorithm engineering competition conceived in Fall 2015 to deepen the relationship between parameterized algorithmics and practice. It aims to:

1. Bridge the divide between the theory of algorithm design 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.
5. Encourage the dissemination of these findings in scientific papers.

In each of the five prior iterations [27, 28, 21, 32, 46] as well as this iteration, participants were asked to provide implementations for one or two specifically chosen problems which provide optimal as well as close to optimal solutions on a given set of selected instances in an appropriate amount of time. In the previous iterations, PACE tackled the problems Treewidth [27, 28], Feedback Vertex Set [27], Minimum Fill-in [28], Steiner Tree [21], Vertex Cover [32], Hypertree width [32], and Treedepth [46]. These challenges have had a significant impact on the research community According to Google Scholar, the previous PACE reports are cited more than 145 times altogether. Moreover, research articles based on concrete implementations competing in previous editions of PACE were published in conferences such as ALENEX, ESA, SEA, and WADS.

In this article, we report on the sixth iteration of PACE. The problem chosen for PACE 2021 is Cluster Editing, also known as Correlation Clustering (see Section 2 for the definition and overview). The challenge featured three tracks. In the *exact track* the goal was to compute optimal solutions for as many instances as possible with a 30-minute time limit per instance. In the *heuristic track* the goal was to compute valid solutions that are as close as possible to being optimal within 10 minutes per instance. In the (new) *kernel track* the goal was twofold: first, to compute an equivalent instance (referred to as kernel) that is as small as possible and, second, to lift valid (but not necessarily optimal) solutions for the kernel to valid solutions for the original instance; we refer to Section 3.1 for a more detailed description of the tracks and their aims.

The PACE 2021 challenge was announced on 22nd October 2020, tracks were specified on 19th November. On 16th December the public instances were made available. From 28th March 2021 on, the participants could test solutions on the public instances via the `optil.io` platform, which provided also a provisional ranking. The final version of the submissions was due on 1st June 2021. Afterwards, the submissions were evaluated on the public as well as a set of hidden instances (see Section 3.2 for details). The results were announced on 16th July 2021. The award ceremony took place during the International Symposium on Parameterized and Exact Computation (IPEC 2021) which was supposed to take place in Lisbon, but due to the pandemic crisis was held online. After the debut with PACE 2020, the current iteration is the second in which short descriptions of the top four solvers in each track are contained as standalone documents in the proceedings of IPEC.

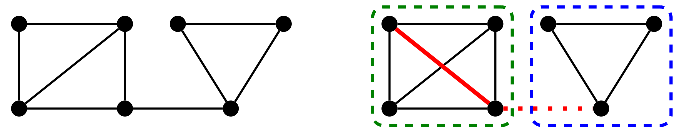

Figure 1 *Left:* An exemplary input graph. *Right:* Two edge modifications (deleting the red dotted edge and adding the thick red edge) suffice to obtain this cluster graph from the input graph. The two clusters are indicated by dashed boxes.

2 Cluster Editing

Graph-based data clustering has numerous applications and there are many approaches to cluster a given graph [58]. Cluster Editing, also known as Correlation Clustering, follows the graph modification approach [4, 9, 59]: Given an undirected graph, the task is to find a minimum-cardinality set of edges to insert or remove in a way that the resulting graph is a cluster graph—a graph where every connected component is a complete graph (called a clique)—see Figure 1 for an example. Herein, the assumption is that a cluster graph gives an ideal clustering: each cluster is maximally connected and no edge exists between two clusters. The graph modification approach lets us find a cluster graph "closest" to the input, that is, a best clustering under the parsimony criterion. One important advantage of this approach is that the number of clusters is not required to be part of the input but is determined implicitly by the input graph. The application fields of Cluster Editing include bioinformatics [9], data mining [4], and psychology [64].

For a given graph $G = (V, E)$ we call a set $S \subseteq \binom{V}{2}$ of vertex pairs a *cluster editing set* if $(V, E \Delta S)$ is a cluster graph, where Δ denotes the symmetric difference.

A graph is a cluster graph if and only if it does not contain a P_3 as an induced subgraph. This characterization gives rise to a simple integer linear programming formulation [41] as well as the following branching strategy: For every induced P_3, add the missing edge to make it a clique, or remove one of the two edges of the P_3 [22]. This results in an algorithm with running time $O(3^k \cdot |V|^3)$, where k is the size of the cluster editing set. A first improvement of this simple branch-and-bound algorithm is due to Gramm et al. [39]; among other results they showed that Cluster Editing is solvable in $O(2.27^k \cdot |V|^3)$ time. Their algorithm combines the P_3 branching strategy with heavy case distinction. The to this date fastest fixed-parameter algorithm with respect to k runs in $O(1.62^k + n + m)$ time [17]. This algorithm uses the so-called *merge branching* technique: When faced with a P_3 induced by the vertices u, v, w, one decides whether or not u and v will end up in the same cluster, and correspondingly merges u and v into a single vertex uv or deletes the edge $\{u, v\}$, respectively. Note that for the merge step one has to introduce edge weights for the edges incident to uv and deal with the special case of weight-0 edges. We remark that all solvers submitted to the exact track solve an integer program or use a branch-and-bound strategy at the heart of their algorithm.

Among many further studies in parameterized algorithmics [11, 18, 34, 44, 48] it was shown that an algorithm with running time subexponential in k, the number of vertices, or the number of edges would refute the exponential time hypothesis (ETH) [45]. Furthermore, Cluster Editing admits polynomial-size kernelizations. Studies in this direction were initialized by Gramm et al. [39], who provided a kernel with $O(k^2)$ vertices. Over the next years the kernel size was improved to $24k$ vertices [33], $4k$ vertices [42], and finally $2k$ vertices [23, 24].

Observe that any cluster editing set is guaranteed to contain at least one edge for every disjoint P_3, so it is natural to ask whether Cluster Editing remains fixed-parameter tractable for the number of edges *above guarantee*. In this line of thought it was shown that Cluster Editing is fixed-parameter tractable with respect to the number of edges above the size of a maximum vertex-disjoint P_3-packing [11], but para-NP-hard with respect to the number of edges above the size of a maximum modification-disjoint P_3-packing [48].

Cluster Editing is also a hot topic in the field of algorithm engineering. There are many heuristic approaches that are empirically shown to provide high-quality solutions, as well as exact algorithms. Most algorithms for the latter combine branch-and-bound strategies with integer linear programming as well as heavy preprocessing [20, 44]. Concerning heuristics for Cluster Editing we would like to highlight two approaches which also inspired some of the submissions to the heuristic track and whose quality was empirically verified. The first is the Louvain method by Blondel et al. [16] which is a greedy hill climbing algorithm initially used for community detection. It tries to maximize the relative density of edges inside the communities compared to those outside. The second approach is the so-called FORCE heuristic [66] in which one interprets the edges and non-edges between the vertices as forces and tries to find vertex positionings which minimize the overall energy in the system. Later, Wittkop et al. [67] combined the FORCE heuristic with a parameterized exact algorithm to obtain higher stability in the solution quality.

3 Challenge Setup

There were three tracks in which the participants could compete: an exact, a heuristic, and a new kernelization (data reduction) track. For each track the 200 instances were selected by the Program Committee (PC), half of them publicly available before the submission deadline. The instances were sorted by increasing (n, m) in lexicographic order, where n is the number of vertices and m the number of edges of the particular instance.

In the testing phase the instances were evaluated on `optil.io` [65]. For the final evaluation, we tested the instances on Intel(R) Xeon(R) CPU E5-1620 3.60 GHz machines using the Linux 4.15 kernel. Both evaluations used the same time limits: 30 minutes for the exact track, 10 minutes for the heuristic track, and 5 minutes for the kernelization track.

3.1 Track Descriptions

The exact and the heuristic track followed essentially the same rules as in previous iterations of PACE. The kernelization track was newly introduced and aimed at shrinking the input as much as possible within a five-minute time limit and return an “equivalent” instance. We subsequently provide the details for each track.

Exact Track. In the exact track submissions had to compute an optimal cluster editing set within 30 minutes for the given instance. While no proof of optimality of the returned cluster editing set is required, we disqualified submissions that returned a suboptimal cluster editing set for some instance (a cluster editing set of strictly smaller size was either known to the PC in advance or computed by other submissions). The optimality testing was conducted also on other instances than the 200 instances of the exact track, including some instances of the heuristic track.

The ranking in the exact track is determined by the number of solved instances with the overall summed running time as a tie breaker if two submissions solved the same number of instances.

Heuristic Track. In the heuristic track submissions had to provide a cluster editing set within 10 minutes for a given instance.

The ranking computation for the heuristic track is inherited from the previous iterations of PACE: For each instance, we collected the minimum size $s_{\min}$ of any found cluster editing set (computed by any submission) and the size s of the cluster editing set computed by the submission. The instance score is then $100 \cdot s_{\min}/s$. For example, a score of 100 indicates the submission found was one of the best for this instance while a score of 50 (25) indicates that the submission found a cluster editing set two (four) times as large as a best known cluster editing set. Overall, the score for each instance is in the interval $[0, 100]$ where a score of 0 was given if no cluster editing set was returned within 10 minutes. The total score is simply the average of the instance scores over the 200 test instances.

Kernel Track. The new kernel track was introduced to evaluate preprocessing techniques for Cluster Editing. The rules are inspired by the kernelization concept, which is arguably among the practically most relevant tools from of parameterized algorithmics [35]. It is defined as follows for decision problems: A kernelization algorithm is a polynomial-time algorithm that, given an instance (I, k) of a parameterized problem L, returns an instance (I', k') such that:

1. (I, k) is *equivalent* to (I', k'), that is $(I, k) \in L \iff (I', k') \in L$, and
2. $|I'| + k' \leq f(k)$ for some computable function f.

Note that there are two apparent issues when we want to apply this concept in practice or in a programming contest:

(a) For many problems (including Cluster Editing) the standard parameter k (solution size) is not known in advance but is to be determined by the respective solver.

(b) Instead of deciding whether there is a cluster editing set of a certain size, the task is usually to compute an optimal cluster editing set.

Our solution to issue (a) is straightforward: For an input graph G for Cluster Editing one returns a number d and a graph G' such that $\mathrm{opt}(G) = \mathrm{opt}(G') + d$; here $\mathrm{opt}(H)$ denotes the size of an optimal cluster editing set for graph H. Our solution to issue (b) is inspired by works on enumeration kernels [10, 25] and lossy kernels [50]: We added the requirement that any submission must provide a so-called *lifting algorithm* which takes a (not necessarily optimal) cluster editing set S' for the kernel, and returns a cluster editing set S' for the original instance such that $|S| \leq |S'| + d$. Note that the latter condition accommodates the fact that suboptimal decisions in S' (over which the submission has no control) can be rectified in the solution lifting algorithm. Since computing $\mathrm{opt}(G')$ involves the potentially very time-consuming task of solving Cluster Editing, we did not strictly verify $\mathrm{opt}(G) = \mathrm{opt}(G') + d$ but instead used several heuristic checks: For the 190 out of 200 instances for which we knew $\mathrm{opt}(G)$, we verified that $\mathrm{opt}(G) \geq d$ and $\mathrm{opt}(G) \geq |S'| + d$. Additionally, we checked $|S| \leq |S'| + d$ for each instance and that the returned set S is indeed a cluster editing set for G (three submissions failed this last test and were disqualified). By using submissions from the heuristic track, we ensured that S' is either optimal or close to being optimal. In hindsight, we consider these heuristic tests to be quite efficient in detecting submissions violating the requirements.

For each instance a submission gets $p = (|V'| + |E'| + 1)/(d+1)$ points, where $G' = (V', E')$ is the graph returned by the kernelization algorithm. Similar to the heuristic track, the instance score is then $100 \cdot p_{\min}/p$, where $p_{\min}$ is the minimum points by any submission.

3.2 Selection of Instances

The exact and kernel track shared their instances, the heuristic track had its own set of instances. The instances were drawn from various sources which we describe below in more detail. Most data sources provided weighted instances, that is, for each pair of vertices there is a number given representing some sort of (dis-)similarity of (or distance between) the two vertices. From such instances we generated multiple unweighted instances by adding edges wherever the corresponding weight was above a certain threshold. More specifically, we proceeded as follows: First all edge weights were linearly scaled to be within the interval $[0, 1]$. Then, for each $t \in \{0.1, 0.2, \ldots, 0.9\}$ we created an unweighted graph by adding an edge $\{u, v\}$ whenever the weight for the vertex pair (u, v) is larger than t. Varying thresholds resulted in instances with a very wide range of difficulty (e.g. from solvable within 1 minute to not solved within 3 hours, by a standard ILP formulation [41] solved with Gurobi). A repository with scripts that download and convert all data is available at `https://github.com/PACE-challenge/Cluster-Editing-PACE-2021-instances`.

The data can be categorized as follows:

Biology This category contains two datasets: a real-world biological dataset[1] that contains COG protein similarity data [55, 19] consisting of 3964 weighted instances of which we chose the 155 instances with between 100 and 5,000 vertices, and a dataset with one weighted instance taken from the data accompanying the TransClust[2] clustering tool [67].

Data Mining This category includes two datasets from which six weighted instances were created. The first dataset is from the World Color Survey[3]; the data is converted based on the descriptions of Regier et al. [56] and Thiel et al. [63] and we created one weighted instance. The second dataset is the newsgroups dataset from scikit-learn[4] [54]; the data is converted based on the descriptions of Thiel et al. [63] and we created five weighted instances.

SNAP This category includes instances found in the SNAP [47] dataset. We took 35 large unweighted graphs having 4,000 up to 2 million vertices. These instances were only used in the heuristic dataset.

Random We used randomly generated data to produce some challenging instances. In particular we randomly created action sequences (sequences of actions performed by a person during computer assisted tests as done e.g. at PIAAC [52]) and converted them into graphs as described by Ulitzsch et al. [64].

For the exact and kernel track we tested our instances with a standard ILP formulation [41] solved with Gurobi. We set a time limit of 3 hours per instance and took the running time as indicator of the difficulty of the instances. In the end, we picked 140 instances that were solved within the 30 minutes, 15 instances that were solved in more than 30 minutes but less than 3 hours, and 45 instances that could not be solved within 3 hours. This resulted in 79 graphs from the Biology category, 13 graphs from the Data Mining category, and 108 graphs from the Random category.

For the heuristic track we picked the data sets such that we had an even distribution with respect to the graph size. This resulted in 84 graphs from the Biology category, 43 graphs from the Data Mining category, 36 graphs from the SNAP category, and 37 graphs from

[1] The dataset is available at `https://bio.informatik.uni-jena.de/data/#cluster_editing_data`.
[2] The dataset is available at `https://transclust.compbio.sdu.dk/main_page/index.php`.
[3] The dataset is available at `http://www.icsi.berkeley.edu/wcs/data.html`
[4] The dataset is available at `https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html`

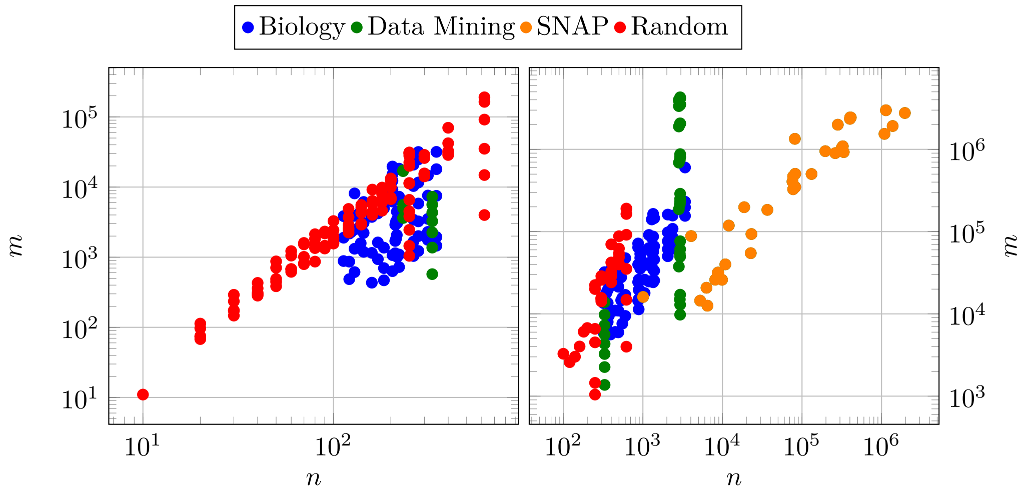

Figure 2 The number of vertices (n) and edges (m) in the two created datasets (left: exact and kernel track; right: heuristic track). In the heuristic track, the first instance with 10 vertices and 31 edges is not shown in order to not clutter the remaining data points too much.

the Random category. Figure 2 displays the number of vertices and edges in the selected instances of the complete dataset.

4 Participants and Results

There were 15, 11, and 6 teams that officially submitted a solution to the exact, heuristic, and kernel track, respectively. Several teams participated in more than one track; in total there were 21 distinct teams with 11 of them being student teams (the implementation is done solely by bachelor / master / PhD students). There were roughly twice as many users that submitted a solution to the `optil.io` server during the testing phase. For each track, the top five on `optil.io` were from participants of PACE 2021. The participants represented three continents and the following 11 countries (number of authors from the respective country is given in brackets): Germany (39), Czechia (6), France (5), Australia (4), India (4), United States (3), Japan (2), Mexico (1), Netherlands (1), Poland (1), and the United Kingdom (1). The results are listed below.

4.1 Exact Track

The ranking for the exact track is listed subsequently; see Figure 3 for an illustration of the performance of the accepted solvers on the full benchmark instances. We list the number of solved instances from the 100 hidden instances and in brackets from the 200 overall instances.

1. Lars Gottesbüren, Tobias Heuer, Thomas Bläsius, Philipp Fischbeck, Michael Hamann, Jonas Spinner, Christopher Weyand, Marcus Wilhelm (Karlsruhe Institute of Technology, Hasso Plattner Institut) solved **87** (171) instances [38].
   `https://github.com/kittobi1992/cluster_editing`
2. Alexander Bille, Dominik Brandenstein, Emanuel Herrendorf (Philipps University of Marburg) solved **81** (160) instances [12].
   `https://github.com/EmanuelHerrendorf/pace-2021`

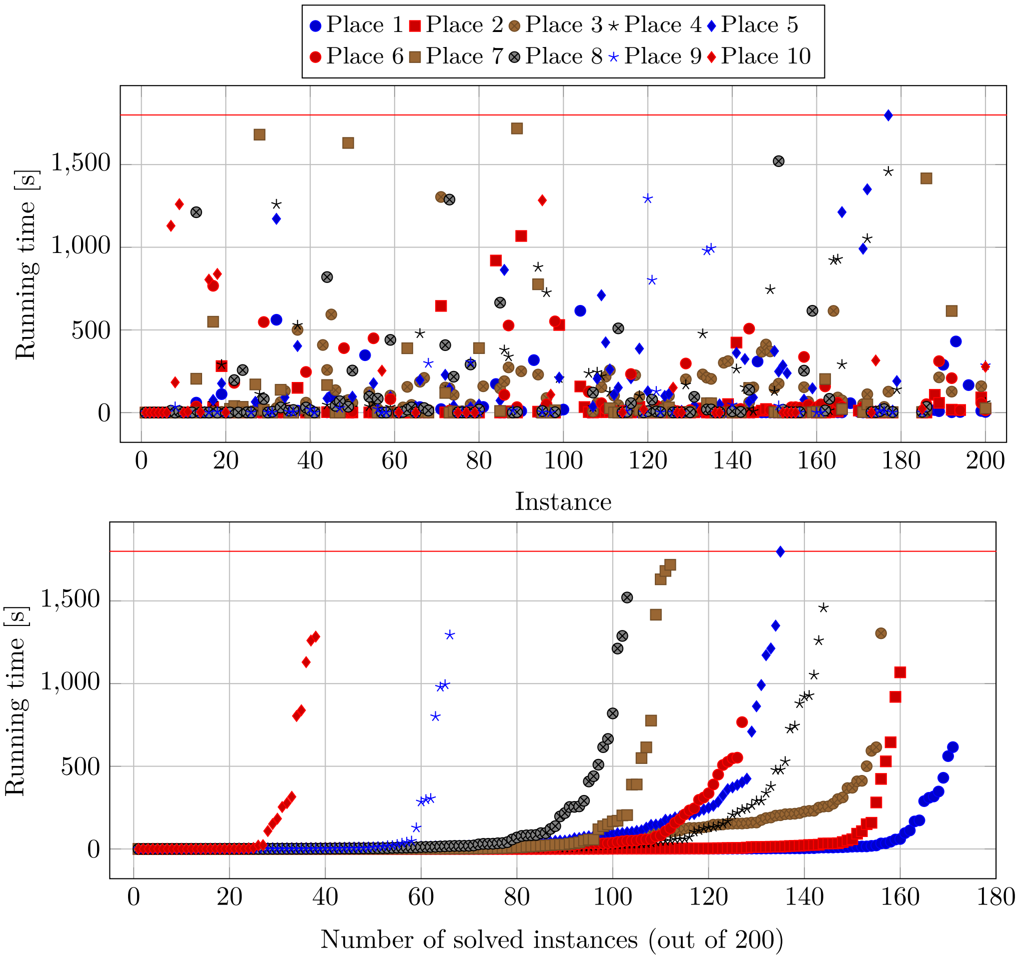

Figure 3 Performance of the top 10 solvers in the exact track. Top: running time plotted for each of the 200 benchmark instances. Bottom: a cactus plot, here a data point with coordinates (x, y) indicates that the corresponding solver could solve x instances of the benchmark set in y seconds per instance. Note that if two solvers solve the same amount of instances within a given time, then the actual set of solved instances can be different (see the top plot). The red horizontal line indicates the timeout of 30 minutes.

3. Valentin Bartier, Gabriel Bathie, Nicolas Bousquet, Marc Heinrich, Théo Pierron, Ulysse Prieto (Grenoble INP, École Normale Supérieure de Lyon, Université de Lyon, University of Leeds) solved **77** (156) instances [7].
`https://github.com/valbart/pace-2021`
4. Jona Dirks, Mario Grobler, Tobias Meis, Roman Rabinovich, Yannik Schnaubelt, Sebastian Siebertz, Maximilian Sonneborn (University of Bremen, Technische Universität Berlin) solved **71** (144) instances [31].
`https://gitlab.informatik.uni-bremen.de/parametrisierte-algorithmen/java/pace-2021-paca-java`
5. Thorben Freese, Jakob Gahde, Mario Grobler, Roman Rabinovich, Fynn Sczuka, Sebastian Siebertz (University of Bremen, Technische Universität Berlin) solved **67** (135) instances [36].
`https://gitlab.informatik.uni-bremen.de/parametrisierte-algorithmen/`

python/paca-python

6. Yosuke Mizutani (University of Utah) solved **63** (127) instances [51].
https://github.com/mogproject/cluster-editing-2021
7. Václav Blažej, Radovan Červený, Dušan Knop, Jan Pokorný, Šimon Schierreich, Ondřej Suchý (Czech Technical University in Prague) solved **59** (112) instances [13].
https://gitlab.fit.cvut.cz/pace-challenge/2021/goat/exact
8. Sachin Agarwal, Sahil Bajaj, Ojasv Singh, Srinibas Swain (IIIT Guwahati) solved **52** (103) instances [2].
https://github.com/sachin-4099/PACE_2021_Cluster_Editing
9. Sebastian Paarmann (Technische Universität Hamburg) solved **36** (66) instances [53].
https://github.com/spaarmann/cluster-editing
10. Tomoki Takayama (Osaka Prefecture University) solved **17** (38) instances [62].
https://github.com/workhouse-lab/pace-2021

- Sylwester Swat (Poznań University of Technology) solved *all* **100** (200) instances but gave suboptimal cluster editing sets on additional test data [61].
https://github.com/swacisko/pace-2021
- Mario Grobler, Roman Rabinovich, Sebastian Siebertz (University of Bremen, Technische Universität Berlin) solved **95** (190) instances but gave suboptimal cluster editing sets on additional test data [40].
https://gitlab.informatik.uni-bremen.de/parametrisierte-algorithmen/cc/pace-2021-paca-cpp
- Moritz Lichter, Oliver Bachtler, Tim Bergner, Irene Heinrich, Alexander Schiewe (TU Darmstadt, TU Kaiserslautern) solved **71** (142) instances but had errors on 5 further instances [49].
https://gitlab.rlp.net/aschiewe/alphabetic
- Kenneth Dietrich, Mario Grobler, Ozan Heydt, Roman Rabinovich, Sebastian Siebertz, Nick Siering, Leon Stichternath, Julian Tat (University of Bremen, Technische Universität Berlin) solved **46** instances but provided suboptimal cluster editing sets on 37 further instances [30].
https://gitlab.informatik.uni-bremen.de/parametrisierte-algorithmen/rust/ceperus/-/tree/v1.0.0

Strategies Used in the Submissions

At the heart of all submissions we find a branch-and-bound algorithm, an ILP solver, or a combination of the two.

All but two submissions (5th and 8th place) use a branch-and-bound approach. At the core of these algorithms is a search tree algorithm that resolves all induced P_3's: this could be a trivial search tree [22], an improved search tree with more case distinctions [39], or the merge branching strategy which is at the core of the theoretically fastest search-tree algorithm [17]. Only Bartier et al. (3rd place) use, to the best of our knowledge, a new branching which starts with each vertex in its own cluster and then merges and reorders clusters; see their solver description for more details. The other submissions (including places 1, 2, and 4 from the top 5) use one of the existing search trees. Even the best theoretical bound on the search-tree size of $O(1.62^k)$ [17] is prohibitively large for e. g. $k \geq 100$ (which is the case in 180 of the 200 instances). Hence, the "bound"-part in the branch-and-bound approach is crucial.

Most submissions employ data reduction rules as well as lower and upper bounds to prune the search tree. There exist various data reduction rules [11, 20, 23, 24, 26, 33, 39, 42], many of which were implemented in several submissions. Interestingly, Gottesbüren et al. (1st place) described new data reduction rules that are apparently very effective; see their solver description for more details. The lower bounds are based on packing disjoint subgraphs. The easiest candidate (included in almost all submissions) is to compute a set $\mathcal{P}$ of modification-disjoint induced P_3's (that is, two P_3's in the packing share at most one vertex) that is as large as possible: any cluster editing set for the instance has size at least $|\mathcal{P}|$ as at least one edge needs to be modified in each P_3 in $\mathcal{P}$. An improvement of this idea is to find packing of subgraphs where more than one edge modification is needed. As an example, Gottesbüren et al. (1st place) and Bartier et al. (3rd place) looked to also include stars in their packing as in each induced $K_{1,\ell}$ at least $\ell - 1$ edges need to be modified. The last type of employed lower bounds is based on the LP-relaxation of the standard ILP formulation, as done by Dirks et al. (4th place). The upper bounds are mostly described in the heuristic track.

The ILP-based approaches work with the standard ILP formulation [41] that has a variable for each possible edge (each vertex pair) and a constraint for each triple of vertices to ensure the resulting graph is P_3-free. Agarwal et al. (8th place) solved the ILP-formulation with the open source solver CBC. Other submissions combined the ILP-solver with initial data reduction, row generation techniques, and the branch-and-bound solver by first measuring some graph parameters and then decide whether to branch or to use the ILP. These approaches were pursued by Bartier et al. (3rd), Dirks et al. (4th), and Freese et al. (5th).

4.2 Heuristic Track

The ranking for the heuristic track based on the 100 hidden instances is as follows (see Figure 4 for an illustration of the performance of the solvers on all 200 benchmark instances):

1. Lars Gottesbüren, Tobias Heuer, Thomas Bläsius, Philipp Fischbeck, Michael Hamann, Jonas Spinner, Christopher Weyand, Marcus Wilhelm (Karlsruhe Institute of Technology, Hasso Plattner Institut) got an average score of **99.9989**/100 [38].
   `https://github.com/kittobi1992/cluster_editing`
2. Sylwester Swat (Poznań University of Technology) got an average score of **99.9985**/100 [61].
   `https://github.com/swacisko/pace-2021`
3. Valentin Bartier, Gabriel Bathie, Nicolas Bousquet, Marc Heinrich, Théo Pierron, Ulysse Prieto (Grenoble INP, École Normale Supérieure de Lyon, Université de Lyon, University of Leeds) got an average score of **99.9975**/100 [6].
   `https://github.com/GBathie/pace_2021_mu_solver`
4. Martin Josef Geiger (University of the Federal Armed Forces Hamburg) got an average score of **99.9876**/100 [37].
   `https://doi.org/10.5281/zenodo.4891323`
5. Emir Demirović (Delft University of Technology) got an average score of **99.9786**/100 [29].
   `https://bitbucket.org/EmirD/pace-2021/`
6. Ben Strasser got an average score of **99.9723**/100 [60].
   `https://github.com/ben-strasser/cluster-editing-pace2021`
7. Angus Ritossa, Paula Tennent, Tiana Tsang Ung, Akshay Valluru (UNSW Sydney) got an average score of **99.8656**/100 [57].
   `https://bitbucket.org/randomsampling/pace21/`
8. Sachin Agarwal, Sahil Bajaj, Ojasv Singh, Srinibas Swain (IIIT Guwahati) got an average score of **99.6739**/100 [3].
   `https://github.com/sahilbajaj82/PACE-2021-Cluster-Editing`

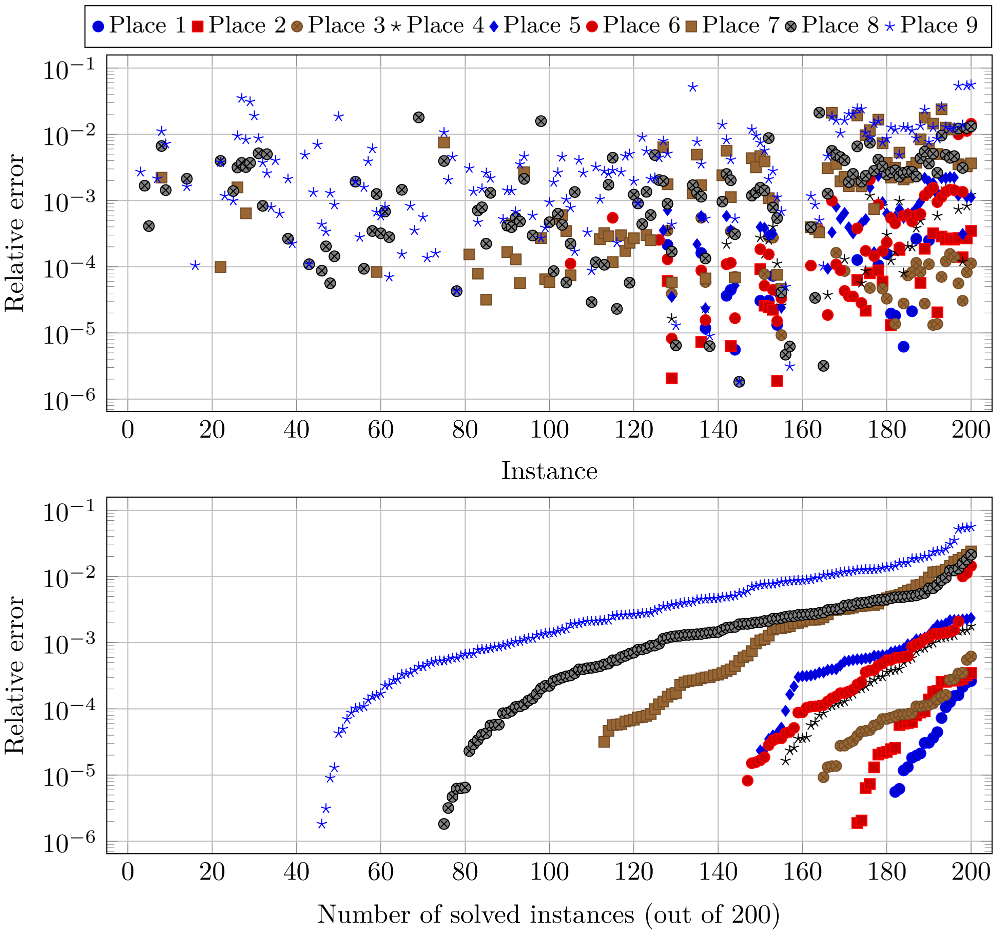

Figure 4 The relative error made by the top nine heuristic submissions (all submissions with an average score higher than 99/100). More precisely, the y-value of a dot is (solution size of submission)/(best known solution size) − 1. In the top plot, the x-axis denotes the respective instance of the benchmark set. In the bottom plot (cactus plot), the x-axis denotes the number of instances where the submission returned a solution with relative error at most the data point's y-value. If a data point is missing (in either plot), then the submission returned a best known solution and the relative error is zero. We remark that the last 20 instances all have solution sizes of more than 300,000 edges (up to 2,500,000 edges). Thus, a relative error of 1% can mean a difference of several thousand edges to the best solution.

9. Václav Blažej, Radovan Červený, Dušan Knop, Jan Pokorný, Šimon Schierreich, Ondřej Suchý (Czech Technical University in Prague) got an average score of **99.4946**/100 [14]. `https://gitlab.fit.cvut.cz/pace-challenge/2021/goat/heuristic`

10. Jona Dirks, Mario Grobler, Tobias Meis, Roman Rabinovich, Yannik Schnaubelt, Sebastian Siebertz, Maximilian Sonneborn (University of Bremen, Technische Universität Berlin) got an average score of **89.0009**/100 [31]. `https://gitlab.informatik.uni-bremen.de/parametrisierte-algorithmen/java/pace-2021-paca-java`

11. Joshua Harmsen and A.J. Zuckerman (Hamilton College) got an average score of **77.1234**/100 [43]. `https://github.com/joshuaharmsen845/PACE-Challenge/tree/sol1`

Strategies Used in the Submissions

Before going into somewhat more details of solution strategies, we discuss a more efficient representation of solutions employed by most submissions. Instead of maintaining sets of edges, one maintains a partition of the vertices with the meaning that each part in the partition forms a clique in the resulting graph. We will refer to the parts in the partition as *clusters*. It is straightforward to translate solutions between these two representations.

All submissions in the top ten incorporate some form of local search. The differences in the submissions were in how much focus was given to the local search part: Some implementations started with a trivial solution and solely focused on the local search part; herein, by trivial solution we mean a solution in which each vertex is in its own cluster or all vertices are in one cluster. Submissions following this strategy include places 1, 3, 4, 5, and 6. We remark that the 3rd placed submission by Bartier et al. first preprocessed the input using data reduction rules. The only other submission employing data reduction rules is by Sylwester Swat (2nd place).

The submissions of places 2, 7, 8, 9, and 10 all used different heuristics to compute the initial solution. Notably, among these submissions, the 2nd placed submission does have the most elaborate local search part. Thus, local search seems overall the most promising heuristic approach to cluster editing and we subsequently describe different options for what local changes were considered by the participants and what were strategies to avoid getting stuck in local optima. The most frequently used local operations are:

1. (The easiest and by far most-frequently used operation.) Moving a vertex v from one cluster C into another cluster C'. Some submissions only consider moving v to clusters C' that contain neighbors of v as these are the only options that could improve the current solution.
2. Putting a vertex v into a newly created cluster; the new cluster then only contains v.
3. Merging two clusters C and C' into one new cluster.
4. Swapping two vertices, that is, removing two vertices from their cluster and adding them to the respective other cluster.

We remark that the list is not exhaustive and variations of the above operations have been employed as well. To avoid getting stuck in local optima, several strategies have been used, that fall broadly on the following two approaches:

1. Restart the computation from scratch. Here the local operations use randomization so it is unlikely to get stuck in the same local optimum.
2. Perform some local changes that do not improve the solution, that is, the cost of the new solution is at least as high as the old solution. These changes could randomly reassign a fixed number or a fraction of vertices to new clusters or temporarily change the cost function for a fixed number of rounds (e.g. temporarily making edge insertions twice as expensive as edge deletions).

In both approaches, the best encountered solution is stored and returned at the end of the program. Notably, the 1st and 3rd place submissions follow the second approach and do not restart computations from scratch.

4.3 Kernel Track

The ranking for the kernel track is as follows:

1. Sylwester Swat (Poznań University of Technology) got an average score of **65.6761**/100 [61].
   `https://github.com/swacisko/pace-2021`

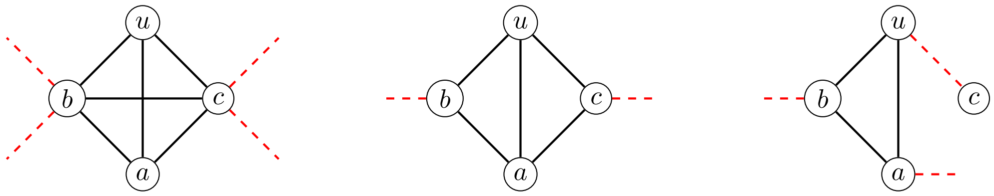

Figure 5 A visualization of the three cases in Reduction Rule 1. The red dashed edges are all present in the input graph and will be removed by the data reduction rule.

- Valentin Bartier, Gabriel Bathie, Nicolas Bousquet, Marc Heinrich, Théo Pierron, Ulysse Prieto (Grenoble INP, École Normale Supérieure de Lyon, Université de Lyon, University of Leeds) got an average score of **71.0077**/100 but their lifting algorithm did not provide valid cluster editing sets on 9 instances [5].
  `https://framagit.org/theo_pierron/pace-2021`
- Václav Blažej, Radovan Červený, Dušan Knop, Jan Pokorný, Šimon Schierreich, Ondřej Suchý (Czech Technical University in Prague) got an average score of **54.0123**/100 but did not provide a lifting algorithm in time (the submission after the deadline passed all our tests) [15].
  `https://gitlab.fit.cvut.cz/pace-challenge/2021/goat/kernelization`
- Moritz Beck, Timon Behr, Johannes Blum, Sabine Cornelsen, Sabine Storandt (University of Konstanz) got an average score of **31.0164**/100 but their lifting algorithm did not provide valid cluster editing sets on 61 instances [8].
  `https://bitbucket.org/moritzbeck/supercereal/`
- Kenneth Dietrich, Mario Grobler, Ozan Heydt, Roman Rabinovich, Sebastian Siebertz, Nick Siering, Leon Stichternath, Julian Tat (University of Bremen, Technische Universität Berlin) got an average score of **26.0103**/100 but their lifting algorithm did not provide a valid cluster editing set on 1 instance [30].
  `https://gitlab.informatik.uni-bremen.de/parametrisierte-algorithmen/rust/ceperus/-/tree/v2.0.0`
- Jona Dirks, Mario Grobler, Tobias Meis, Roman Rabinovich, Yannik Schnaubelt, Sebastian Siebertz, Maximilian Sonneborn (University of Bremen, Technische Universität Berlin) got an average score of **18.0**/100 but did not provide a lifting algorithm [31].
  `https://gitlab.informatik.uni-bremen.de/parametrisierte-algorithmen/java/pace-2021-paca-java`

Strategies Used in the Submissions

We briefly discuss the data reduction techniques used in the submissions to the kernel track. Note that many submissions from the exact and also some from the heuristic track also employ a subset of these techniques. Various submissions employ (subsets of) existing data reduction rules [11, 20, 23, 24, 26, 33, 39, 42]. While refraining from listing these established rules, let us mention two new rules employed in the submissions. Bartier et al. (kernel track) provided the following rule that deals with low degree vertices occurring in a triangle (see Figure 5 for an illustration).

▶ **Reduction Rule 1** (Bartier et al.). Let u be a vertex with neighborhood $\{a, b, c\}$ and let $L = \{a, b, c, u\}$.

- If the vertices in L induce a K_4, a has degree three, and b and c both have degree at most 5, then isolate L. Here, isolating L means removing all edges with exactly one endpoint in L and reducing k accordingly.

- If the vertices in L induce a diamond (a K_4 minus one edge) and a, b, and c have all degree at most three, then isolate L.
- If $G[\{a, b, c\}]$ contains only the edge $\{a, b\}$ and a and b have all degree at most three, then isolate $\{a, b, u\}$.

Gottesbüren et al. (1st place in exact track) also provided some additional data reduction rules. Among these, the following rule using lower and upper bounds, while simple, proved particularly effective.

▶ **Reduction Rule 2** (Gottesbüren et al.)**.** If modifying an edge e would raise the lower bound above the current upper bound, then e is not allowed to be modified.

Of course Reduction Rule 2 highly depends on the used upper and lower bounds. However, it would be interesting to see whether this or a similar rule could be used to show a problem kernel with respect to some above-guarantee parameterization (recall that the number k of edge modifications is rather large on the benchmark instances).

5 PACE Organization

The Program Committee of PACE 2021 consisted of André Nichterlein (chair), Leon Kellerhals, Tomohiro Koana, and Philipp Zschoche, all from Technische Universität Berlin. During the organization of PACE 2021 the Steering Committee (SC) was composed of

Édouard Bonnet	LIP, ENS Lyon,
Holger Dell	Goethe University Frankfurt and IT University of Copenhagen,
Johannes Fichte	Technische Universität Dresden,
Markus Hecher	Technische Universität Wien,
Bart M. P. Jansen (chair)	Eindhoven University of Technology,
Łukasz Kowalik	University of Warsaw,
Marcin Pilipczuk	University of Warsaw, and
Manuel Sorge	Technische Universität Wien.

In July 2021, André Nichterlein joined the SC, while Édouard Bonnet left. The Program Committee of PACE 2022 will be chaired by Christian Schulz (University of Heidelberg).

6 Conclusion

We thank all participants for their enthusiasm and impressive work and look forward to PACE 2022. We hope that future iterations will again feature a kernel track to further push the development of data reduction rules and kernelization algorithms.

We welcome anyone who is interested to add their name to the mailing list on the website `https://pacechallenge.org/` to receive PACE updates and join the discussion. For frequent updates, especially for updates on plans for PACE 2022, also see the `@pace_challenge` Twitter account.

References

1. Networks project, 2017. URL: `http://www.thenetworkcenter.nl`.
2. Sachin Agarwal, Sahil Bajaj, Ojasv Singh, and Srinibas Swain. Cluster editing using ILP (CLIP): An exact solver for cluster editing, 2021. URL: `https://github.com/sachin-4099/PACE_2021_Cluster_Editing`.

3 Sachin Agarwal, Sahil Bajaj, Ojasv Singh, and Srinibas Swain. Conflict reduced best-fit cluster (CoRBeC): A heuristic solver for cluster editing, 2021. URL: `https://github.com/sahilbajaj82/PACE-2021-Cluster-Editing`.

4 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004. `doi:10.1023/B:MACH.0000033116.57574.95`.

5 Valentin Bartier, Gabriel Bathie, Nicolas Bousquet, Marc Heinrich, Théo Pierron, and Ulysse Prieto. PACE 2021 kernelization track, 2021. `doi:10.5281/zenodo.4947841`.

6 Valentin Bartier, Gabriel Bathie, Nicolas Bousquet, Marc Heinrich, Théo Pierron, and Ulysse Prieto. Pace 2021 muSolver, 2021. `doi:10.5281/zenodo.4947325`.

7 Valentin Bartier, Gabriel Bathie, Nicolas Bousquet, Marc Heinrich, Théo Pierron, and Ulysse Prieto. valbart/pace-2021: PACE 2021 exact track, 2021. `doi:10.5281/zenodo.4935569`.

8 Moritz Beck, Timon Behr, Johannes Blum, Sabine Cornelsen, and Sabine Storandt. Super Cereal, 2021. `doi:10.5281/zenodo.4892806`.

9 Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3-4):281–297, 1999. `doi:10.1089/106652799318274`.

10 Matthias Bentert, Till Fluschnik, André Nichterlein, and Rolf Niedermeier. Parameterized aspects of triangle enumeration. *Journal of Computer and System Sciences*, 103:61–77, 2019. `doi:10.1016/j.jcss.2019.02.004`.

11 René van Bevern, Vincent Froese, and Christian Komusiewicz. Parameterizing edge modification problems above lower bounds. *Theory of Computing Systems*, 62(3):739–770, 2018. `doi:10.1007/s00224-016-9746-5`.

12 Alexander Bille, Dominik Brandenstein, and Emanuel Herrendorf. PACE 2021 exact track submission, 2021. `doi:10.5281/zenodo.4889012`.

13 Václav Blažej, Radovan Červený, Dušan Knop, Jan Pokorný, Šimon Schierreich, and Ondřej Suchý. GOAT, 2021. URL: `https://gitlab.fit.cvut.cz/pace-challenge/2021/goat/exact`.

14 Václav Blažej, Radovan Červený, Dušan Knop, Jan Pokorný, Šimon Schierreich, and Ondřej Suchý. GOAT, 2021. URL: `https://gitlab.fit.cvut.cz/pace-challenge/2021/goat/heuristic`.

15 Václav Blažej, Radovan Červený, Dušan Knop, Jan Pokorný, Šimon Schierreich, and Ondřej Suchý. GOAT, 2021. URL: `https://gitlab.fit.cvut.cz/pace-challenge/2021/goat/kernelization`.

16 Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: Theory and Experiment*, 10, 2008. `doi:10.1088/1742-5468/2008/10/P10008`.

17 Sebastian Böcker. A golden ratio parameterized algorithm for cluster editing. *Journal of Discrete Algorithms*, 16:79–89, 2012. `doi:10.1016/j.jda.2012.04.005`.

18 Sebastian Böcker and Jan Baumbach. Cluster editing. In *Proceedings of the 9th Conference on Computability in Europe (CiE 2013)*, volume 7921 of *LNCS*, pages 33–44. Springer, 2013. `doi:10.1007/978-3-642-39053-1_5`.

19 Sebastian Böcker, Sebastian Briesemeister, Quang Bao Anh Bui, and Anke Truß. Going weighted: Parameterized algorithms for cluster editing. *Theoretical Computer Science*, 410(52):5467–5480, 2009. `doi:10.1016/j.tcs.2009.05.006`.

20 Sebastian Böcker, Sebastian Briesemeister, and Gunnar W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60(2):316–334, 2011. `doi:10.1007/s00453-009-9339-7`.

21 Édouard Bonnet and Florian Sikora. The PACE 2018 parameterized algorithms and computational experiments challenge: The third iteration. In *Proceedings of the 13th International Symposium on Parameterized and Exact Computation (IPEC '18)*, volume 115 of *LIPIcs*, pages 26:1–26:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.IPEC.2018.26`.

22 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.

23 Yixin Cao and Jianer Chen. Cluster Editing: Kernelization based on edge cuts. *Algorithmica*, 64(1):152–169, 2012.

24 Jianer Chen and Jie Meng. A $2k$ kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 78(1):211–220, 2012.

25 Nadia Creignou, Arne Meier, Julian-Steffen Müller, Johannes Schmidt, and Heribert Vollmer. Paradigms for parameterized enumeration. *Theory of Computing Systems*, 60(4):737–758, 2017. doi:10.1007/s00224-016-9702-4.

26 Lucas de O. Bastos, Luiz Satoru Ochi, Fábio Protti, Anand Subramanian, Ivan César Martins, and Rian Gabriel S. Pinheiro. Efficient algorithms for cluster editing. *Journal of Combinatorial Optimization*, 31(1):347–371, 2016. doi:10.1007/s10878-014-9756-7.

27 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 *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC '16)*, volume 63 of *LIPIcs*, pages 30:1–30:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.30.

28 Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC '17)*, volume 89 of *LIPIcs*, pages 30:1–30:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.IPEC.2017.30.

29 Emir Demirović. Kanpai: A bottom-up approach for cluster editing (PACE 2021), 2021. URL: https://bitbucket.org/EmirD/pace-2021/.

30 Kenneth Dietrich, Mario Grobler, Ozan Heydt, Roman Rabinovich, Sebastian Siebertz, Nick Siering, Leon Stichternath, and Julian Tat. PACA-RUST, 2021. doi:10.5281/zenodo.4884693.

31 Jona Dirks, Mario Grobler, Tobias Meis, Roman Rabinovich, Yannik Schnaubelt, Sebastian Siebertz, and Maximilian Sonneborn. PACA-JAVA, 2021. doi:10.5281/zenodo.4884681.

32 M. Ayaz Dzulfikar, Johannes Klaus Fichte, and Markus Hecher. The PACE 2019 parameterized algorithms and computational experiments challenge: The fourth iteration (invited paper). In *Proceedings of the 14th International Symposium on Parameterized and Exact Computation (IPEC '19)*, volume 148 of *LIPIcs*, pages 25:1–25:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.IPEC.2019.25.

33 Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Peter Shaw. Efficient parameterized preprocessing for Cluster Editing. In *Proceedings of the 16th International Symposium on Fundamentals of Computation Theory (FCT '07)*, volume 4639 of *LNCS*, pages 312–321. Springer, 2007. doi:10.1007/978-3-540-74240-1_27.

34 Fedor V. Fomin, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of Cluster Editing with a small number of clusters. *Journal of Computer and System Sciences*, 80(7):1430–1447, 2014.

35 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

36 Thorben Freese, Jakob Gahde, Mario Grobler, Roman Rabinovich, Fynn Sczuka, and Sebastian Siebertz. PACA-PYTHON, 2021. doi:10.5281/zenodo.4884234.

37 Martin Josef Geiger. Source code for PACE 2021, 2021. doi:10.5281/zenodo.4891323.

38 Lars Gottesbüren, Tobias Heuer, Thomas Bläsius, Philipp Fischbeck, Michael Hamann, Jonas Spinner, Christopher Weyand, and Marcus Wilhelm. KaPoCE - an exact and heuristic solver for the cluster editing problem, 2021. doi:10.5281/zenodo.4892524.

39 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, 38(4):373–392, 2005.

40 Mario Grobler, Roman Rabinovich, and Sebastian Siebertz. PACE 2021 - cluster editing, c++, 2021. `doi:10.5281/zenodo.4889505`.

41 Martin Grötschel and Yoshiko Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1-3):59–96, 1989. `doi:10.1007/BF01589097`.

42 Jiong Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8-10):718–726, 2009. `doi:10.1016/j.tcs.2008.10.021`.

43 Joshua Harmsen and A.J. Zuckerman. Cluster editing with existing cliques, 2021. URL: `https://github.com/joshuaharmsen845/PACE-Challenge/tree/sol1`.

44 Sepp Hartung and Holger H. Hoos. Programming by optimisation meets parameterised algorithmics: a case study for cluster editing. In *Proceedings of the 9th International Conference on Learning and Intelligent Optimization, LION 2015*, volume 8994 of *LNCS*, pages 43–58. Springer, 2015. `doi:10.1007/978-3-319-19084-6_5`.

45 Christian Komusiewicz and Johannes Uhlmann. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics*, 160(15):2259–2270, 2012.

46 Lukasz Kowalik, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Manuel Sorge, and Piotr Wygocki. The PACE 2020 parameterized algorithms and computational experiments challenge: Treedepth. In *Proceedings of the 15th International Symposium on Parameterized and Exact Computation (IPEC '20)*, volume 180 of *LIPIcs*, pages 37:1–37:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.IPEC.2020.37`.

47 Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection, June 2014. URL: `http://snap.stanford.edu/data/index.html`.

48 Shaohua Li, Marcin Pilipczuk, and Manuel Sorge. Cluster editing parameterized above modification-disjoint P_3-packings. In *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS '21)*, volume 187 of *LIPIcs*, pages 49:1–49:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.STACS.2021.49`.

49 Moritz Lichter, Oliver Bachtler, Tim Bergner, Irene Heinrich, and Alexander Schiewe. PACE solver description – Alphabetic, 2021. URL: `https://gitlab.rlp.net/aschiewe/alphabetic`.

50 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC '17)*, pages 224–237. ACM, 2017. `doi:10.1145/3055399.3055456`.

51 Yosuke Mizutani. PACE 2021 - exact, 2021. `doi:10.5281/zenodo.4877899`.

52 OECD. Technical report of the survey of adult skills (PIAAC). Technical report, Paris, France, 2013. URL: `https://www.oecd.org/skills/piaac/_Technical%20Report_17OCT13.pdf`.

53 Sebastian Paarmann. Pandora cluster editing solver, 2021. `doi:10.5281/zenodo.4964394`.

54 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

55 Sven Rahmann, Tobias Wittkop, Jan Baumbach, Marcel Martin, Anke Truss, and Sebastian Böcker. Exact and heuristic algorithms for weighted cluster editing. In *Proceedings of the 6th Computational Systems Bioinformatics Conference (CSB '07)*, pages 391–401. World Scientific, 2007. `doi:10.1142/9781860948732_0040`.

56 Terry Regier, Paul Kay, and Naveen Khetarpal. Color naming reflects optimal partitions of color space. *Proceedings of the National Academy of Sciences*, 104(4):1436–1441, 2007. `doi:10.1073/pnas.0610341104`.

57 Angus Ritossa, Paula Tennent, Tiana Tsang Ung, and Akshay Valluru. PACE challenge 2021 solver description, from the random sampling group of the UNSW GraphAbility VIP project, 2021. `doi:10.5281/zenodo.4946084`.

58 Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. doi:10.1016/j.cosrev.2007.05.001.

59 Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004. doi:10.1007/3-540-36379-3_33.

60 Ben Strasser. CELMEC: Cluster editing pace 2021, 2021. URL: https://github.com/ben-strasser/cluster-editing-pace2021.

61 Sylwester Swat. CluES - a heuristic solver for the cluster editing problem, 2021. URL: https://github.com/swacisko/pace-2021.

62 Tomoki Takayama. SatoxaSolver: A submission for PACE 2021, 2021. doi:10.5281/zenodo.4941172.

63 Erik Thiel, Morteza Haghir Chehreghani, and Devdatt P. Dubhashi. A non-convex optimization approach to correlation clustering. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '19), the 31st Innovative Applications of Artificial Intelligence Conference (IAAI '19), and the 9th AAAI Symposium on EducationalAdvances in Artificial Intelligence (EAAI '19)*, pages 5159–5166. AAAI Press, 2019. doi:10.1609/aaai.v33i01.33015159.

64 Esther Ulitzsch, Qiwei He, Vincent Ulitzsch, Hendrik Molter, André Nichterlein, Rolf Niedermeier, and Steffi Pohl. Combining clickstream analyses and graph-modeled data clustering for identifying common response processes. *Psychometrika*, 86(1):190–214, 2021. doi:10.1007/s11336-020-09743-0.

65 Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. Optil.io: Cloud based platform for solving optimization problems using crowdsourcing approach. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, CSCW '16 Companion, page 433–436. ACM, 2016. doi:10.1145/2818052.2869098.

66 Tobias Wittkop, Jan Baumbach, Francisco P. Lobo, and Sven Rahmann. Large scale clustering of protein sequences with force — a layout based heuristic for weighted cluster editing. *BMC Bioinformatics*, 8(396), 2007. doi:10.1186/1471-2105-8-396.

67 Tobias Wittkop, Dorothea Emig, Sita Lange, Sven Rahmann, Mario Albrecht, John H Morris, Sebastian Böcker, Jens Stoye, and Jan Baumbach. Partitioning biological data with transitivity clustering. *Nature Methods*, 7(6):419–420, 2010. URL: https://doi.org/10.1038/nmeth0610-419.