PACE Solver Description: The KaPoCE Exact Cluster Editing Algorithm*

Thomas Bläsius
Karlsruhe Institute of Technology, Germany

Philipp Fischbeck
Hasso Plattner Institute, Potsdam, Germany

Lars Gottesbüren
Karlsruhe Institute of Technology, Germany

Michael Hamann
Karlsruhe Institute of Technology, Germany

Tobias Heuer
Karlsruhe Institute of Technology, Germany

Jonas Spinner
Karlsruhe Institute of Technology, Germany

Christopher Weyand
Karlsruhe Institute of Technology, Germany

Marcus Wilhelm
Karlsruhe Institute of Technology, Germany

Abstract

The cluster editing problem is to transform an input graph into a cluster graph by performing a minimum number of edge editing operations. A cluster graph is a graph where each connected component is a clique. An edit operation can be either adding a new edge or removing an existing edge. In this write-up we outline the core techniques used in the exact cluster editing algorithm of the KaPoCE framework (contains also a heuristic solver), submitted to the exact track of the 2021 PACE challenge.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases cluster editing

Digital Object Identifier 10.4230/LIPIcs.IPEC.2021.27

Supplementary Material *Software (Source Code)*: `https://doi.org/10.5281/zenodo.4892524`
Software (Source Code): `https://github.com/kittobi1992/cluster_editing`

1 Preliminaries

Let $G = (V, E)$ be a simple, undirected graph. The cluster editing problem asks to transform G into a disjoint union of cliques with the least number of edge edit operations. An edit is the deletion of an existing edge or the insertion of a missing edge. As a graph is a cluster-graph if and only if it does not contain an induced path on three vertices (a P_3), the problem can also be seen as P_3-free editing.

2 Solver Summary

The solver is based on a more general $\mathcal{F}$-free solver developed for quasi-threshold editing, i.e., $\{C_4, P_4\}$-free editing [6]. Its implementation already solves more than half of the public exact instances in the 2021 PACE challenge without modification. The solver uses a branch-and-bound algorithm that branches on the edges of a forbidden subgraph. In our adaptation, branching is simplified following the work of recent FPT algorithms [3, 2] by generalizing to the weighted cluster editing problem where each vertex pair has an associated integral edit cost. To find the lowest number of edits, the solver actually uses the decision variant that asks if it is possible to solve the instance with k edits. The optimization problem is solved

* This is a brief description of one of the highest ranked solvers of PACE Challenge 2021. It has been made public for the benefit of the community and was selected based on the ranking. PACE encourages publication of work building on the ideas presented in this description in peer-reviewed venues.

by calling the decision variant with increasing values of k. The solver includes a multitude of known reduction rules [4].

Initially, we split the input graph into connected components and solve them individually. After applying all possible reduction rules, the solver branches on the most expensive edge. The edge is marked either as permanent or the vertex pair is forbidden from being connected. Future edits are prevented from changing the state of the vertex pair by setting its edit cost to infinity. As outlined in [3], the endpoints of permanent edges are merged to obtain an equivalent weighted instance of smaller size.

Crucial for the performance of a branch-and-bound algorithm are good lower and upper bounds to prune branches of the search tree that cannot lead to better solutions. Moreover, they enable highly effective parameter-dependent reduction rules [4].

3 Upper Bounds

An initial upper bound is obtained with our heuristic solver. Surprisingly, this initial solution is optimal on all public instances we were able to solve.

4 Lower Bounds

We extend the lower bound from P_3 packings used in the $\mathcal{F}$-free solver [6] to packings of arbitrary structures. We found stars to be particularly effective because they circumvent the problem that the lower bound from P_3 packings cannot be higher than $|E|/2$. For a detailed explanation of lower bounds via subgraph packing and related techniques (e.g., local search) we refer to the $\mathcal{F}$-free solver [6].

5 Reduction Rules

The solver uses the reduction rules 1,2,3 and 5 from [4]. Rule 4, which is based on min-cuts, was found to be ineffective. We implemented the unweighted $4k$ kernel based on critical cliques from [7] and the weighted $2k$ kernel from [5] which is the smallest known kernel for the problem. However, the $2k$ kernel is not used in the final solver since it is mostly dominated by the other reductions and cannot always be applied when dealing with zero-cost edges.

Additionally, the following four reduction rules are used.

Distance Three Reduction

Two vertices with distance three or more cannot be in the same cluster in an optimal solution [1]. Therefore, all vertex pairs with distance three or more are initially marked as forbidden. This does not apply to weighted instances however.

Forced Choices Reduction for all Vertex Pairs

The most effective reduction rule we added is based on lower and upper bounds. If setting an edge to forbidden or permanent would raise the lower bound above the current upper bound, then the opposite edit must be performed. In other words, we identify an edge where, if branched on it, one branch would be pruned immediately.

A naive implementation of this rule is too slow as it requires a quadratic number of lower bound (i.e. packing) computations. However all these packings are similar. Given a packing lower bound for the instance, we locally modify the packing for each vertex pair to obtain the required bounds. Because a packing changes only locally, this can be done significantly

faster than computing n^2 packing lower bounds from scratch. Note that this rule dominates previously known parameter-dependent reduction rules from [4].

Forced Choices Reduction for one Vertex Pair

The locally modified packings from the above reduction are smaller than a heuristically found packing with local search. Thus, we additionally identify a constant number of edits that are highly unlikely to be included in the optimal solution and compute lower bounds for all these edits from scratch. Due to the high quality of the heuristic upper bounds, these two packing based reductions alone are able to solve most instances without branching.

Clique-Like Subgraph Reduction

We analyze the clusters found by the heuristic solver. Using an exact solver as a subroutine for a cluster and its neighborhood, we identify some clusters that are present in an optimal solution. Such clusters can be removed from the initial instance.

References

1 Lucas 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.

2 Sebastian Böcker. A golden ratio parameterized algorithm for cluster editing. *Journal of Discrete Algorithms*, 16:79–89, 2012.

3 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.

4 Sebastian Böcker, Sebastian Briesemeister, and Gunnar W Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60(2):316–334, 2011.

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

6 Lars Gottesbüren, Michael Hamann, Philipp Schoch, Ben Strasser, Dorothea Wagner, and Sven Zühlsdorf. Engineering Exact Quasi-Threshold Editing. In *18th International Symposium on Experimental Algorithms (SEA 2020)*, volume 160, pages 10:1–10:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.SEA.2020.10.

7 Jiong Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8-10):718–726, 2009.