The 'Pandora' Cluster Editing Solver

Sebastian Paarmann @
Technische Universität Hamburg, Germany

Abstract

'Pandora' is a solver for the Cluster Editing problem developed for the PACE Challenge 2021. It is based on a fixed-parameter bounded search tree algorithm and multiple data reduction rules.

2012 ACM Subject Classification Applied computing; Mathematics of computing → Graph algorithms; Mathematics of computing → Solvers

Keywords and phrases Cluster Editing, fixed-parameter algorithms, graph theory, PACE challenge

Supplementary Material Source code available on Github and on Zenodo
Software: `https://github.com/spaarmann/cluster-editing` [7]

Acknowledgements I want to thank Prof. Dr. Matthias Mnich and Dr. Jens M. Schmidt who were my advisors on the Bachelor thesis that resulted in this work.

1 Introduction

This document provides a brief description of the techniques employed in creating the 'Pandora' solver for the Cluster Editing problem. It is part of the submission of the solver to the PACE Challenge 2021. The solver was written as part of a Bachelor thesis in Computer Science, and a much more detailed description of most of the solver is also available in form of the thesis itself, although some improvements have been made to the solver after the thesis was finalized.

As a brief introduction, the Cluster Editing problem asks to transform an arbitrary unweighted, undirected input graph into a *cluster graph*. A cluster graph is a graph in which every connected component forms a clique, i.e. it consists only of one or more disjoint cliques. This transformation should be achieved by producing a set of edges to either insert or delete from the graph, and when solving the problem exactly, the goal is to find a *minimal* set of edits that results in a cluster graph.

Formally, for an input graph $G = (V, E)$ the goal is to find an $F \subseteq \binom{V}{2}$ such that $(V, E \Delta F)$ is a cluster graph and $|F|$ is minimal.

2 Bounded Search Tree Algorithm

Internally, the input problem instance is converted into an equivalent instance for the Weighted Cluster Editing problem. Here, the graph is characterized by a weight (or *cost*) function $s\colon \binom{V}{2} \to \mathbb{Z}$, such that $uv \in E \Leftrightarrow s(uv) > 0$. For an edge $uv \in E$, $s(uv)$ are the deletion costs of the edge, while for a pair $uv \notin E$, $-s(uv)$ are the insertion costs. The cost of a cluster editing F is then $\sum_{e \in F} |s(e)|$ and the goal to find a cluster editing with minimal cost.

The solver implements a *fixed-parameter algorithm*, where the problem has an additional parameter k which limits the allowed cost of a solution. Thus, for an input (G, k) it should return a solution F if a solution with cost less than or equal k exists, or 'no solution' otherwise. To solve the original optimization problem, the algorithm is simply executed repeatedly with increasing k to find the minimal k for which a solution can be found.

The basic structure of the algorithm is a *bounded search tree*. The search tree strategy implemented was introduced by Böcker et. al [2] and briefly repeated here: We choose an

edge $uv \in E$ to branch and take two branches: Forbidding uv and *merging* uv. To forbid uv we reduce k by $s(uv)$ and set $s(uv) = -\infty$, deleting the edge from the graph. Merging uv means removing both u and v from the graph and replacing them by a new vertex u', setting $s(u'w) = s(uw) + s(vw) \forall w \in V \setminus \{u, v\}$. For $s(uw) \neq -s(vw)$, k can be reduced by $\min\{|s(uw)|, |s(vw|\}$. For $s(uw) = s(vw)$, the resulting edge $u'w$ has weight 0 which requires some additional care; this is described in more detail in the original paper by Böcker et al. After merging u and v are considered to have a permanent edge between them.

At every branching step we choose the edge $uv \in E$ that minimizes the *branching number* of the corresponding branching step. The branching number can be calculated from the branching vector (a, b) where a is the cost of deleting uv and b is the cost of merging uv. For a description of how a branching number can be derived from the branching vector, see e.g. [5]. In practice we have pre-computed a polynomial function that approximates the branching number from the branching vector to allow quick computation. This branching strategy results in a search tree of size $O(2^k)$.

Before each branching step we also calculate a simple lower bound for the remaining cost. If that lower bound exceeds the current parameter k, we can immediately return 'no solution' for this branch of the search tree.

3 Reduction Rules

The solver also implements multiple *reduction rules* which can reduce the size or complexity of a problem instance by essentially proving that certain edits must be part of an optimal solution (in general, or in the current part of the search tree).

The first reduction is based on the idea of *critical cliques* in the unweighted input graph:

▶ **Definition 1.** *A* critical clique K *is an induced clique in* G *where all vertices of* K *have the same set of neighbors outside of* K*, and* K *is maximal under this property.*

Guo proved that an optimal cluster editing will never split a critical clique into multiple clusters [6]. As a first step in the algorithm, for every critical clique K, we can merge all the vertices of K into a single vertex, transforming the unweighted input into an equivalent weighted one that the remainder of the algorithm works on.

We also implement a set of parameter-independent reductions that are applied once at the start, and then again at regular intervals on the intermediate instances in the search tree. These were described by Böcker et al. in [3].

Rule 1 *(heavy non-edge rule)* Forbid a non-edge uv with $s(uv) < 0$ if

$$|s(uv)| \geq \sum_{w \in N(u)} s(uw).$$

Rule 2 *(heavy edge rule, single end)* Merge vertices u, v of an edge uv if

$$s(uv) \geq \sum_{w \in V \setminus \{u,v\}} |s(uw)|.$$

Rule 3 *(heave edge rule, both ends)* Merge vertices u, v of an edge uv if

$$s(uv) \geq \sum_{w \in N(u) \setminus \{v\}} s(uw) + \sum_{w \in N(v) \setminus \{u\}} s(vw).$$

Rule 4 *(almost clique rule)* Let k_C be the min-cut value of $G[C]$, for $C \subseteq V$. The vertices of C can be merged if

$$k_C \geq \sum_{u,v \in C, s(uv) \leq 0} |s(uv)| + \sum_{u \in C, v \in V \setminus C, s(uv) > 0} s(uv).$$

Rule 5 *(similar neighborhood)* Merge uv if

$$s(uv) \geq \max_{C_u, C_v} \min\{s(v, C_v) - s(v, C_u) + \Delta_v, s(u, C_u) - s(u, C_v) + \Delta_u\} \tag{1}$$

with the maximum running over all $C_u, C_v \subseteq W$ with $C_u \cap C_v = \emptyset$.

Böcker et al. [1] also introduced a parameter-dependent reduction that takes into account the current parameter k. We first define for each pair uv the *induced costs* of forbidding or merging it:

$$\mathsf{icf}(uv) = \sum_{w \in N(u) \cap N(v)} \min\{s(uw), s(vw)\}$$

$$\mathsf{icp}(uv) = \sum_{w \in (N(u) \Delta N(v)) \setminus \{u,v\}} \min\{|s(uw)|, |s(vw)|\}$$

One can then show that the following reduction rules are correct:

- For $u, v \in V$ where $\mathsf{icf}(uv) + \max\{0, s(uv)\} + b(G, uv) > k$: Merge u and v
- For $u, v \in V$ where $\mathsf{icp}(uv) + \max\{0, -s(uv)\} + b(G, uv) > k$: Forbid uv

where $b(G, uv)$ is a lower bound on the cost of an optimal solution for solving $G - \{u, v\}$.

And finally we also implemented a reduction rule derived from a kernelization based on edge cuts introduced by Cao and Cen [4]. The original kernelization and reduction only works for graphs that do not contain any edges with weight 0, however as we mentioned before, such edges can be created by the merging procedure. We thus only apply the reduction once at the beginning, before any zero edges can have been produced.

First, for a vertex $v \in V$ define the *deficiency* $\delta(v) = \sum_{x,y \in N(v), xy \notin E} -s(xy)$ and for a set $S \subseteq V$ define $\gamma(S) = \sum_{x \in S, y \in V \setminus S, xy \in E} s(xy)$. Then the *stable cost* of v is defined as $\rho(v) = 2\delta(v) + \gamma(N[v])$ and $N[v]$ is *reducible* if $\rho(v) < |N[v]|$. If $N[v]$ is reducible, we can insert any missing edges in $G[N[v]$ to make it a clique, reducing k appropriately. Additionally, for each $x \in N(N[v])$, if the total weight of edges between x and $N[v]$ is less than or equal to the weight of all non-edges between x and $N[v]$, then we can delete all edges between x and $N[v]$ and reduce k appropriately.

References

1 Sebastian Böcker, Sebastian Briesemeister, Quang Bao Anh Bui, and Anke Truß. A fixed-parameter approach for weighted cluster editing. In Alvis Brazma, Satoru Miyano, and Tatsuya Akutsu, editors, *Proceedings of the 6th Asia-Pacific Bioinformatics Conference, APBC 2008, 14-17 January 2008, Kyoto, Japan*, volume 6 of *Advances in Bioinformatics and Computational Biology*, pages 211–220. Imperial College Press, 2008. URL: `http://www.comp.nus.edu.sg/%7Ewongls/psZ/apbc2008/apbc050a.pdf`.

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

3 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`.

4 Yixin Cao and Jianer Chen. Cluster editing: Kernelization based on edge cuts. *Algorithmica*, 64(1):152–169, 2012. doi:10.1007/s00453-011-9595-1.

5 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

6 Jiong Guo. A more effective linear kernelization for cluster editing. *Theor. Comput. Sci.*, 410(8-10):718–726, 2009. doi:10.1016/j.tcs.2008.10.021.

7 Sebastian Paarmann. Pandora cluster editing solver, June 2021. doi:10.5281/zenodo.4964394.