



PACE Solver Description: Cluster Editing Kernelization Using CluES*

Sylwester Swat  

Institute of Computing Science, Poznań University of Technology, Poland

Abstract

This article briefly describes the most important algorithms and techniques used in the cluster editing kernelization solver called “CluES”, submitted to the 6th Parameterized Algorithms and Computational Experiments Challenge (PACE 2021).

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Cluster editing, kernelization, graph algorithms, PACE 2021

Digital Object Identifier 10.4230/LIPIcs.IPEC.2021.35

Supplementary Material *Software (Source Code)*: <https://doi.org/10.5281/zenodo.4949787>

1 Problem description

A cluster editing set of a graph $G = (V, E)$ is a set of edges $E_{mod} \subseteq V \times V$ such that each connected component of a graph $G[E \Delta E_{mod}]$ is a clique, where $A \Delta B$ denotes the symmetric difference of sets A and B . In the cluster editing problem we aim to find a cluster editing set of smallest possible size. In the cluster editing kernelization problem we aim to find a set $X \subset V \times V$, with size as large as possible, such that $X \subset E_{opt}$ for some optimal solution E_{opt} to the cluster editing problem.

2 Solver description

In this paper we provide a short description of the most important algorithms implemented in solver CluES. Due to many parameters used in the implementation and a vast number of case distinctions that need to be taken into account, this description may not contain full information about the algorithms behavior in every possible situation.

Though there are many kernelization rules implemented in CluES (see section 3), the resulting set X of edge modifications returned by the solver is found based on a set of heuristically found cluster editing sets: given cluster editing sets S_1, \dots, S_p we find the set $X = \bigcap_{i=1}^p S_i$ and return it as a final result. It is necessary to mention that the result found this way need not be correct, as the set X may not be contained in any optimum cluster editing set of graph G . Set X will not be a correct result only if all sets S_i contain the same pair $(u, v) \in V \times V$ that does not belong to any optimum solution. By creating a large number of high-quality cluster editing sets and trying to avoid finding similar local optima multiple times, we decrease the probability of returning an incorrect answer. It may also happen that all sets S_i are optimal cluster editing sets, but we will get $X = \emptyset$. In that case, as a resulting set, we can take the largest set of edge modification found by valid kernelization rules described in section 3.

* 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.



© Sylwester Swat;

licensed under Creative Commons License CC-BY 4.0

16th International Symposium on Parameterized and Exact Computation (IPEC 2021).

Editors: Petr A. Golovach and Meirav Zehavi; Article No. 35; pp. 35:1–35:3

Leibniz International Proceedings in Informatics



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

In the algorithm we take $p = 20$ and for each $i \leq p$ we find the set S_i using the heuristic solver CluES (article “PACE Solver Description: CluES - a heuristic solver for the cluster editing problem” in the current proceedings). We need to add here that sets S_i are found using many different sets of parameters that usually enable us to find cluster editing sets of very high quality. What is more important, by selecting appropriate values for parameters we are often able to find many different local optima to the cluster editing problem. Even if those local optima are not optimal solutions from the global point of view, they are often optimum in some small (local) subgraphs of given graph G . By steering the parameters we try to avoid getting stuck in similar local optima and thus we reduce the risk of returning an incorrect answer set X .

3 Kernelization rules

We use over 15 parameter-independent kernelization rules based on edge cuts and critical cliques. Additionally we use several preprocessing procedures that are not valid kernelization rules (there is no guarantee of preserving the optimality of the solution size after performing those procedures), but are just a reasonable guess that often enables us to quickly and accurately identify nodes that most often belong to the same cluster in some good solution. Those heuristic procedures are also very useful for finding different local optima.

3.1 Valid kernelization rules

Nine of implemented valid kernelization rules are based on concepts related to critical cliques and can be found in [3] and [2].

Next valid kernelization rule is a “similar neighborhoods” rule (see [1]) applied to an unweighted graph. The rule states that, if there exists an edge $(u, v) \in E$ with $N[u] = N[v] \cup \{w\}$ for some $w \notin N[v]$, then there exists an optimal solution with u and v belonging to the same cluster.

Another three valid kernelization rules are specifications of the “almost clique rule” [1] applied for all triangles, K_4 ’s and induced diamonds. The “almost clique rule” states that, if there exists a subset $C \subset V$ such that $G[C]$ is connected and the size of the minimum cut in $G[C]$ is not smaller than $|\{(a, b) \in C \times C : (a, b) \notin E\}| + |\{(a, b) \in E : a \in C, b \notin C\}|$, then there exists an optimal solution where all nodes in C belong to the same cluster.

3.2 Heuristic preprocessing procedures

Now, we proceed to describe heuristic preprocessing procedures. Most of them are based on properties of some intersections of neighborhoods of critical cliques. The following rules depend on miscellaneous parameters and are usually executed multiple times for different values of those parameters.

1. Let $A = \{K_1, \dots, K_p\}$ be a set of critical cliques in G with the same neighborhood $N(K_i)$ for all $i \leq p$ and let $G' = G[V \setminus \bigcup_{i=1}^p K_i]$. If there exists an index $i \leq p$ and a set $K' \subset V$ such that K' is a critical clique in G' , $N(K_i) \subset K'$, $|N(K_i)| \leq \alpha \cdot |K'|$ and $|K_i| \leq \beta \cdot |K'|$, where α and β are parameters, then make K_i a single cluster and remove it from G .
2. Let $A = \{u_1, \dots, u_p\}$ be a set of nodes for which $N(u_i)$ is a clique and let $G' = G[V \setminus A]$. If there exists an index $i \leq p$ and a set $K \subset V$ such that K is a critical clique in G' , $N(u_i) \subset K$ and $|N(u_i)| \leq \alpha \cdot |K|$, where α is a parameter, then make u_i a single cluster and remove it from G .

3. Let $A = \{K_1, \dots, K_p\}$ be a set of critical cliques in G for which $N(K_i)$ is a clique and $\beta_1 \leq |K_i| \leq \beta_2$, where β_1, β_2 are parameters. Take $G' = G[V \setminus \bigcup_{i=1}^p K_i]$. If there exists an index $i \leq p$ and a set $K' \subset V$ such that K' is a critical clique in G' , $N(K_i) \subset K'$ and $|N(K_i)| \leq \alpha \cdot |K'|$, where α is a parameter, then make K_i a single cluster and remove it from G . This rule is a generalization of the previous rule (the previous rule can be obtained by setting $\beta_1 = \beta_2 = 1$).
4. Let $A = \{u_1, \dots, u_p\}$ be a set of nodes with the same neighborhood $N(u_i)$. Let $m = \alpha \cdot |A|$, where α is a parameter. Add edges (u_{2i-1}, u_{2i}) for $1 \leq i \leq m$ to graph G . It is important to note here that all added edges are stored and, if at some stage of the main algorithm a cluster editing set S_i is found with a single cluster containing both ends of a stored edge, then the edge is removed from S_i .
5. Let $(u, v) \in E$ be an edge with $N[u] \subset N[v]$, $|N[u]| \geq \alpha \cdot |N[v]|$, $|N(v)| - |N(u)| \leq \beta$ and $|N(u) \cap N(v)| \geq \gamma$. Mark nodes u, v to belong to the same cluster. Marking nodes is done by creating a proper permutation graph before finding cluster editing sets using a heuristic solver CluES. By setting $\alpha = \gamma = 0$, $\beta = 1$ we can obtain the “similar neighborhoods” rule described in 3.1.
6. Let $C = \{v_1, \dots, v_p\} \subset V$ be a set of nodes for which $G[C]$ is connected. For $v \in C$ let $deg_{out}(v) = |N(v) \setminus C|$. Assume that for parameters α , β and γ the following conditions hold:
 - a. $\max_{v \in C} deg_{out}(v) \leq \alpha$
 - b. $\sum_{v \in C} deg_{out}(v) \leq \beta$
 - c. $\max_{u, v \in C} |(N(u) \cap N(v)) \setminus C| \leq \gamma$

Then mark all nodes $v \in C$ with $deg_{out}(v) \leq \delta$ (where δ is a parameter) to belong to the same cluster and, depending on values of the parameters, remove from the graph some edges from a set $\{(u, v) \in E : u \in C, v \notin C\}$. This procedure is applied to all triangles, K_4 's and induced diamonds. It is also applied to some clusters found by the heuristic solver CluES. Let us mention that knowing the structure of the graph $G[C]$ and selecting proper values of parameters we can obtain the “almost clique rule” described in 3.1.

4 Availability

The source code of CluES is available at <https://doi.org/10.5281/zenodo.4949787> (please select the proper version of the solver to access required contest track: heuristic, exact or kernelization).

References

- 1 Sebastian Böcker, Sebastian Briesemeister, and Gunnar Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60:316–334, January 2008.
- 2 Jianer Chen and Jie Meng. A 2k kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 78(1):211–220, 2012. JCSS Knowledge Representation and Reasoning.
- 3 Jiong Guo. A more effective linear kernelization for cluster editing. *Theor. Comput. Sci.*, 410:718–726, March 2009.