Cluster Editing with Locally Bounded Modifications Revisited

Peter Damaschke

Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
ptr@chalmers.se

Abstract. For Cluster Editing where both the number of clusters and the edit degree are bounded, we speed up the kernelization by almost a factor n compared to Komusiewicz and Uhlmann (2012), at cost of a marginally worse kernel size bound. We also give sufficient conditions for a subset of vertices to be a cluster in some optimal clustering.

1 Introduction

We consider graphs $G = (V, E)$ with n vertices and m edges and use standard notation like $N(v)$ for the open neighborhood of a vertex, $N[v] := N(v) \cup \{v\}$, $N[W] := \bigcup_{v \in W} N[v]$, and $N(W) := N[W] \setminus W$. A *cluster graph* is a disjoint union of cliques. G is a cluster graph if and only if G has no *conflict triple*, i.e., three vertices inducing exactly two edges. Cluster Editing asks to turn a graph $G = (V, E)$ into a cluster graph using at most k edge *edits*, that is, insertions or deletions. An *optimal clustering* is an optimal solution to Cluster Editing. The *edit degree* of v is the number of edits v is incident to. Cluster Editing has applications in, e.g., molecular biology, and is well studied [1, 2, 5, 9]. Cluster Editing is NP-hard [10], but is also fixed-parameter tractable (FPT) in k. The currently smallest problem kernel has $2k$ vertices [4, 3]. An obstacle for the practical use of an FPT algorithm is that k may still be too large. Thus, stronger parameters have been proposed:

(d, t)-Constrained Cluster Editing [6]: Given a graph $G = (V, E)$, parameters d, t, and k, and individual constraints $\tau(v) \leq t$ for every vertex v, transform G into a cluster graph with at most d clusters, by applying at most k edits, such that every vertex v has edit degree at most $\tau(v)$.

The problem is in FPT already in the combined parameter (d, t). A kernel with at most $4dt$ vertices is computed in $O(n^3)$ time [6]. An $O(dt)$ kernel can be preferable to an $O(k)$ kernel: If every vertex actually has edit degree $\Theta(t)$, we get $k = \Theta(nt)$, whereas d is typically much smaller than n. Besides kernel sizes, the time for the kernelization is also decisive for scalability. Here we speed up the kernelization for (d, t)-Constrained Cluster Editing. We compute a kernel with at most $5dt + d$ vertices in $O((m + dt^2) \log n)$ time, and randomization yields a kernel size arbitrarily close to $4dt$ in $O(m + dt^2 \log n)$ expected time.

Note that $m = O(n^2)$, and $dt < n$ can be assumed. Hence our time is within $O(n^2 \log n)$, with a sliightly worse kernel size bound. This appears to be a good deal. Moreover, $O((m+dt^2)\log n)$ is within a logarithmic factor of the size (edge number m) of the graph. Namely, assuming $dt < n$, we have $dt^2 < nt < n^2/d$, and a graph of d disjoint cliques has $\Omega(n^2/d)$ edges. Another difference to [6] is that the "old" reduction rules are only based on the (d,t)-constraints, whereas we find edit-optimal clusters, as defined below.

Definition 1. *In a graph $G = (V, E)$, a subset $C \subseteq V$ is an* edit-optimal cluster *if G has an optimal solution to* Cluster Editing *where C is one of the clusters.*

The value of this notion is the following: Once some C is recognized as edit-optimal, we can safely split off C and work on $G - C$ in order to solve Cluster Editing on G. Of course, we cannot expect a polynomial algorithm to recognize edit-optimal clusters, because iterated application would solve Cluster Editing in polynomial time. Therefore we quest for *sufficient* conditions π: If some C satisfies π, then C is edit-optimal. Ideally, π should be both efficient and not too restrictive. Here is a known example of a condition π used in a kernelizations: Let $\gamma(C)$ be the number of edges between C and $G - C$, and let $\delta(C)$ be the number of non-edges in C. If some $v \in C$ satisfies $N[v] = C$, and $2\delta(C) + \gamma(C) < |C|$, then C is edit-optimal [3]. This inequality relates the total number of edits to $|C|$. Here we will add further conditions π that relate only the edit degrees and list edge coloring numbers to $|C|$. Note that we only study the combinatorial side. Further research could deal with algorithmic implications, e.g., the complexity of recognizing such edit-optimal clusters, and perhaps smaller kernels.

Pointing out optimal clusters in parts of a large graph can be more appropriate than computing an optimal clustering of the entire graph. For instance, it has been observed [7] that large social networks tend to consist of a core that lacks a clear clustering structure, and a periphery containing clusters, which are "exotic" groups that are highly connected but have little external interaction.

2 Faster Kernelization

Definition 2. *Let $C \subset V$ be a vertex set in a graph $G = (V, E)$. The edit degree of $v \in C$ is the number of edits incident to v when we split off C as a cluster. The edit degree of C is the maximum edit degree of the vertices in C.*

Lemma 1. *Let v be any fixed vertex, and g the degree of v. Assume that some cluster $C \ni v$ with edit degree at most t exists. Let $c := |C|$, and let $m(C)$ denote the number of edges incident to C. If $g > 4t$ or $c \geq g > 3t$, then C is uniquely determined, and we can compute C in $O((m(C) + t^2)\log c)$ time.*

Proof. Define $x := |N(v) \setminus C|$ and $y := |C \setminus N(v)|$. Since $v \in C$ is incident to at most t edits, we have $x + y \leq t$. Also note that $|C \cap N(v)| = g - x = c - y$.

Every vertex $u \in C$ is incident to at most t non-edges in C, hence u has at least $g - t - x$ neighbors in $N(v)$, even in $C \cap N(v)$. Every vertex $u \notin C$ has at

most $t+x$ neighbors in $N(v)$ (namely, at most t in $C \cap N(v)$, and at most x in $N(v) \setminus C$). In the case $g > 4t$ we have $t+x \leq 2t < g/2 = g-g/4-g/4 \leq g-t-x$. In the case $c \geq g > 3t$ (hence $x \leq y$ and $x \leq t/2$) we have $t+x \leq t+(x+y)/2 \leq t+t/2 = 3t/2 < g/2 \leq g-3t/2 \leq g-t-t/2 \leq g-t-x$. Thus, by comparing $|N(u) \cap N(v)|$ with $g/2$ we can decide whether $u \notin C$ or $u \in C$, in these two cases. We refer to this comparison as the *threshold test* for u. It also follows that $C \ni v$ with edit degree at most t is uniquely determined in the mentioned cases.

Next we show how to calculate C, or recognize that v is in no cluster with edit degree at most t. Since $|C \setminus N(v)| \leq t$, every vertex in C can have at most $2t$ neighbors outside $N(v)$.

Phase 1: We do the threshold test for all $u \in N(v)$ as follows. We traverse the adjacency list of every such u and check for each neighbor of u whether it also belongs to $N(v)$. Actually we can stop as soon as (i) $g/2$ neighbors in $N(v)$ or (ii) $2t+1$ neighbors outside $N(v)$ are found, or (iii) the end of u's adcacency list is reached. In case (i) and (iii) we know $u \in C$ and $u \notin C$, respectively. In case (ii), v is in no cluster with edit degree at most t. Using a dictionary for $N(v)$, membership of a vertex in $N(v)$ can be tested in $O(\log g)$ time.

Phase 2: We also find the at most t vertices of $C \setminus N(v)$ as follows. Since they all have neighbors in $C \cap N(v)$, it suffices to do the threshold test for the $O(ct)$ vertices in $N(C \cap N(v))$ only. More precisely, we traverse the adjacency lists of all vertices in $C \cap N(v)$ again and count how often every vertex $u \notin N(v)$ appears there. According to our threshold test, we have $u \in C$ if and only if u is met at least $g/2$ times. Each time a vertex u is met, u can be retrieved (for incrementing its count) in $O(\log c + \log t)$ time.

All edges in the adjacency lists of vertices $u \in C \cap N(v)$, processed in Phase 1, are incident to C. We also have to deal with vertices $u \in N(v) \setminus C$, but there are at most t of them, and in each of their lists we consider at most $(t-1)+(2t+1)$ edges which are not incident to C: those ending in $N(v) \setminus C$ again, and at most $2t+1$ edges ending outside $N(v)$. (After that we can stop, as we saw above.) All edges in the lists processed in Phase 2 are also incident to C. Hence $O(t^2)$ processed edges are not indicent to C. Since $g = O(c)$ and $t = O(c)$, each of $\log g$, $\log c$, $\log t$ is $O(\log c)$, thus every edge is processed in $O(\log c)$ time. □

Theorem 1. *We can compute a kernel for* (d,t)-Constrained Cluster Editing *with at most* $5dt+d$ *vertices in* $O((m+dt^2)\log n)$ *time.*

Proof. As long as there exists some vertex v of degree $g > 4t$ we apply Lemma 1 to find $C \ni v$ with edit degree at most t. If no such C exists, the problem instance has no solution. If C exists, it is uniquely determined due to Lemma 1. We check whether the individual degree constraints $\tau(u)$ are satisfied by all $u \in C$, remove C and all incident edges, and reduce the individual degree constraints of vertices outside C by the respective numbers of adjacent vertices in C. If the test in C fails or a constraint drops below zero, the instance has no solution. By iteration we get rid of all vertices of degree above $4t$, and every set C we have obtained must be a cluster in any valid solution. Any vertex of degree $g \leq 4t$ must belong to a cluster of size $c \leq 5t+1$, since $c \leq g+t+1$ holds in general. Since at most

d clusters are allowed, there remain at most $5dt + d$ vertices, or we know that there is no solution. By Lemma 1, every cluster C is found in $O((m(C)+t^2)\log c)$ time. Since edges incident to some C are processed only once, the $m(C)$ terms sum up to at most m. The t^2 term appears at most d times. □

This kernel consists of vertices of degree at most $4t$ in at most d clusters of size at most $5t + 1$. Using Lemma 1 we can keep on trying vertices v with $4t \geq g > 3t$. If v belongs to a cluster of size $c \geq g$ and edit degree t, this appears in a valid solution, by the same reasoning as in Theorem 1. However, trying all $O(dt)$ vertices would require $O((m + dt^3)\log c)$ time. This situation suggests the idea of picking v randomly. The following time bound holds with high probability, since the probability not to hit a large cluster decreases exponentially over time.

Theorem 2. *We can compute a kernel for* (d, t)-Constrained Cluster Editing *with at most* $(4 + \epsilon)dt$ *vertices in* $O((m + (1/\epsilon)dt^2)\log n)$ *expected time.*

3 Sufficient Conditions for Edit-Optimal Clusters

Theorem 3. *Every* C *with* $c := |C| \geq 5t$ *is an edit-optimal cluster.*

The proof replaces any clustering $C_1, \ldots, C_d$ with $C, C_1 \setminus C, \ldots, C_d \setminus C$ (splitting off C as a new cluster) and shows by bookkeeping arguments that the number of edits is not increased. – Compare Theorem 3 to condition $2\delta(C)+\gamma(C) < c$ from [3]. We get that already $t = O(c)$ is sufficient, whereas the total number of edits can be tc, and even quadratic in c. We also connect this result to Section 2. Suppose that we have computed a kernel for (d, t)-Constrained Cluster Editing, but we may also want to solve unrestricted Cluster Editing, because it might need fewer edits. The size bounds in Section 2 remain valid if we remove only those edit-optimal clusters C with $c > 5t$. This yields:

Corollary 1. *If a graph has a clustering with at most* d *clusters, where every vertex has edit degree at most* t*, then we can compute a kernel for* Cluster Editing *with at most* $5dt$ *vertices in* $O((m + dt^2)\log n)$ *time.*

Next we derive a further condition related to another graph problem:

List Edge Coloring: Given a graph where every edge carries a list of suitable colors, paint every edge such that incident edges get distinct colors.

Definition 3. *Given a vertex set* $C \subset V$ *in a graph* $G = (V, E)$*, we define an auxiliary graph* $H(C)$ *on vertex set* $N[C]$*, where the edges of* $H(C)$ *are the non-edges in* C *and the cut edges between* C *and* $G - C$*, that is, the edits incident to* C *when we split off* C *as a cluster. Let every vertex* $s \in C$ *represent a* color*.*

Theorem 4. *Consider the graph* $H(C)$*, colors corresponding to the vertices of* C*, and let* $s \in C$ *be a suitable color for edge* uv *if and only if* s *is not adjacent to* u, v *in* $H(C)$*. Then the following condition is sufficient for* C *being an edit-optimal cluster:* $H(C)$ *admits a list edge coloring with the extra property that, if an edge* e *uses a vertex of edge* f *as its color, then edge* f *may not use a vertex of edge* e *as its color.*

The proof is sketched as follows: When an edge uv is colored with s, then u, s, v form a conflict triple with an edge in C. Every edit in $H(C)$ is in such a conflict triple, conversely, every such conflict triple contains exactly one edit. The conditions enforce that no two of our conflict triples share two vertices. Since, in any clustering, some pair from every conflict triple must be edited, splitting off C as a cluster requires the minimum number of edits incident to C.

This result provides further possibilities to confirm edit-optimal clusters. As a little example, let C be an isolated clique K_7 from which the 6 edges of some K_4 are removed. Then $2\delta(C) + \gamma(C) = 2 \cdot 6 + 0 = 12 > 7$, also $t = 3$, hence the previous conditions do not apply. But the 6 edges of $H(C)$ can be colored by 3 suitable colors represented by vertices of C. The algorithmic use of our coloring condition needs to be further explored. There is a well-known close connection between degrees and edge colorings (see [8]), and progress on the List Edge Coloring problem could further strengthen our conditions.

Acknowledgments: This work has been supported by project grants from the Swedish Research Council (Vetenskapsrådet), 2010-4661, "Generalized and fast search strategies for parameterized problems", and the Swedish Foundation for Strategic Research (SSF), "Data-driven secure business intelligence".

References

1. Böcker, S.: A Golden Ratio Parameterized Algorithm for Cluster Editing. J. Discr. Algor. 16, 79–89 (2012)
2. Böcker, S., Briesemeister, S., Klau, G.W.: Exact Algorithms for Cluster Editing: Evaluation and Experiments. Algorithmica 60, 316–334 (2011)
3. Cao, Y., Chen, J.: Cluster Editing: Kernelization Based on Edge Cuts. Algorithmica 64, 152–169 (2012)
4. Chen, J., Meng, J.: A $2k$ Kernel for the Cluster Editing Problem. J. Comp. System Sci. 78, 211–220 (2012)
5. Fomin, F.V., Kratsch, S., Pilipczuk, M., Pilipczuk, M., Villanger, Y.: Tight Bounds for Parameterized Complexity of Cluster Editing. In: Portier, N., Wilke, T. (eds.) STACS 2013. LIPIcs, vol. 20, pp. 32–43, Dagstuhl (2013)
6. Komusiewicz, C., Uhlmann, J.: Cluster Editing with Locally Bounded Modifications. Discr. Appl. Math. 160, 2259–2270 (2012)
7. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Math. 6, 29–123 (2009)
8. Misra, J., Gries, D.: A Constructive Proof of Vizing's Theorem. Info. Proc. Letters 41, 131–133 (1992)
9. Rahmann, S., Wittkop, T., Baumbach, J., Martin, M., Truß, A., Böcker, S.: Exact and Heuristic Algorithms for Weighted Cluster Editing. In: Markstein, P., Xu, Y. (eds.) CSB 2007, pp. 391–401, Imperial College Press (2007)
10. Shamir, R., Sharan, R., Tsur, D. Cluster Graph Modification Problems. Discr. Appl. Math. 144, 173–182 (2004)