Kernelization Through Tidying

A Case Study Based on s-Plex Cluster Vertex Deletion*

René van Bevern, Hannes Moser, and Rolf Niedermeier

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
{rene.bevern,hannes.moser,rolf.niedermeier}@uni-jena.de

Abstract. We introduce the NP-hard graph-based data clustering problem s-Plex Cluster Vertex Deletion, where the task is to delete at most k vertices from a graph so that the connected components of the resulting graph are s-plexes. In an s-plex, every vertex has an edge to all but at most $s-1$ other vertices; cliques are 1-plexes. We propose a new method for kernelizing a large class of vertex deletion problems and illustrate it by developing an $O(k^2s^3)$-vertex problem kernel for s-Plex Cluster Vertex Deletion that can be computed in $O(ksn^2)$ time, where n is the number of graph vertices. The corresponding "kernelization through tidying" exploits polynomial-time approximation results.

1 Introduction

The contributions of this work are two-fold. On the one hand, we introduce a vertex deletion problem in the field of graph-based data clustering. On the other hand, we propose a novel method to derive (typically polynomial-size) problem kernels for NP-hard vertex deletion problems whose goal graphs can be characterized by forbidden induced subgraphs. More specifically, using "kernelization through tidying", we provide a quadratic-vertex problem kernel for the NP-hard s-Plex Cluster Vertex Deletion problem, for constant $s \geq 1$.

***s*-Plex Cluster Vertex Deletion.** Many vertex deletion problems in graphs can be considered as "graph cleaning procedures", see Marx and Schlotter [11]. This view particularly applies to graph-based data clustering, where the graph vertices represent data items and there is an edge between two vertices iff the two items are similar enough [14]. Then, a cluster graph is a graph where every connected component forms a cluster, a dense subgraph such as a clique in the most extreme case. Due to faulty data or outliers, the given graph may not be a cluster graph and it needs to be cleaned in order to become a cluster graph. A recent example for this is the NP-hard Cluster Vertex Deletion problem [9], where the task is to delete as few vertices as possible such that the resulting graph is a disjoint union of cliques. In contrast, in the also NP-hard s-Plex Cluster Vertex Deletion problem we replace cliques with s-plexes (where s is typically a small constant):

* Supported by the DFG, project AREG, NI 369/9.

Published in the Proceedings of the 9th Latin American Theoretical Informatics Symposium (LATIN'10)*, Oaxaca, México, April 2010. Volume 6034 in Lecture Notes in Computer Science, pages 528–539.*

s-Plex Cluster Vertex Deletion (s-PCVD)
Input: An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Is there a vertex set $S \subseteq V$ with $|S| \leq k$ such that $G[V \setminus S]$ is a disjoint union of s-plexes?

Herein, an *s-plex* is a graph where every vertex has an edge to all but at most $s-1$ other vertices [13]. Subsequently, we refer to the solution set S as *s-plex cluster graph vertex deletion set (s-pvd set)*. The concept of s-plexes has recently received considerable interest in various fields, see, e.g., [2,6,7,12]. The point of replacing cliques with s-plexes in the context of cluster graph generation is that s-plexes better allow to balance the number of vertex deletions against the sizes and densities of the resulting clusters. Note that too many vertex deletions from the input graph may too strongly change the original data whereas asking for cliques as clusters often seems overly restrictive [5,13]. Summarizing, s-PCVD blends and extends previous studies on Cluster Vertex Deletion [9] (which is the same as 1-PCVD) and on s-Plex Editing [7], where in the latter problem one adds and deletes as few *edges* as possible to transform a graph into an s-plex cluster graph.

Problem kernelization. Data reduction and problem kernelization is a core tool of parameterized algorithmics [8]. Herein, viewing the underlying problem as a decision problem, the goal is, given any problem instance x (a graph in our case) with a parameter k (the number of vertex deletions in our case), to transform it in polynomial time into a new instance x' with parameter k' such that $|x'|$ is bounded by a function in k (ideally, a polynomial in k), $k' \leq k$, and (x, k) is a yes-instance iff (x', k') is a yes-instance. We call (x', k') the *problem kernel*. It is desirable to get the problem kernel size $|x'|$ as small as possible. By means of a case study based on s-PCVD, we will present a method to develop small problem kernels for vertex deletion problems where the goal graph can be characterized by a set of forbidden induced subgraphs. For instance, if the goal graph shall be a disjoint union of cliques, then it is characterized by forbidding induced P_3's [14], that is, induced paths containing three vertices. A more complex characterization has been developed for graphs that are disjoint unions of s-plexes [7]. We term our data reduction approach "kernelization through tidying"—it uses a polynomial-time constant-factor approximation to "tidy up" the graph to make data reduction rules applicable.

Discussion of results. Complementing and extending results for Cluster Vertex Deletion [9], we prove an $O(k^2 s^3)$-vertex problem kernel for s-PCVD, which can be computed in $O(ksn^2)$ time. Note that the related edge modification problem s-Plex Editing has an $O(ks^2)$-vertex problem kernel which can be computed in $O(n^4)$ time [7]. We emphasize that the underlying kernelization algorithms are completely different in both cases and that vertex deletion is a "more powerful" operation than edge modification, so a larger problem kernel in the case of s-PCVD does not come unexpectedly. Our main conceptual contribution is the "kernelization through tidying" method outlined in Section 2. There is related work by Kratsch [10] that provides polynomial-size problem kernels for constant-factor approximable problems contained in the classes MIN $\mathrm{F}^{+}\Pi_1$ and

MAX NP. The s-PCVD problem is contained in MIN $\mathrm{F}^{+}\Pi_1$. Applying Kratsch's more general method to s-PCVD would lead to an $k^{O(s)}$-vertex kernel. By way of contrast, in our $O(k^2s^3)$-vertex bound, the value of s does not influence the degree of the polynomial in k, a significant advantage. Other related work is due to Abu-Khzam [1], who considers problem kernels for Hitting Set problems. Again, translating our problem instances into Hitting Set instances (which can be done in a straightforward way) and applying a kernelization for Hitting Set would yield problem instances which are size-bounded by polynomials whose degree depends on s.

Due to space limits, most proofs are deferred to a full version of this paper.

Notation. We only consider *undirected* graphs $G = (V, E)$, where V is the set of vertices and E is the set of edges. Throughout this work, we use $n := |V|$ and $m := |E|$. The *open neighborhood* $N(v)$ of a vertex $v \in V$ is the set of vertices that are adjacent to v. For a vertex set $U \subseteq V$, we define $N(U) := \bigcup_{v \in U} N(v) \setminus U$. We call a vertex $v \in V$ *adjacent* to $V' \subseteq V$ if v has a neighbor in V'. Analogously, we call $U \subseteq V$ *adjacent* to a vertex set $W \subseteq V$ with $W \cap U = \emptyset$ if $N(U) \cap W \neq \emptyset$. We call two vertices v and w *connected* in G if there exists a path from v to w in G. For a set of vertices $V' \subseteq V$, the *induced subgraph* $G[V']$ is the graph over the vertex set V' with the edge set $\{\{v, w\} \in E \mid v, w \in V'\}$. For $V' \subseteq V$, we use $G - V'$ as an abbreviation for $G[V \setminus V']$. For a set $\mathcal{F}$ of graphs, we call a graph $\mathcal{F}$-free if it does not contain any graph from $\mathcal{F}$ as an induced subgraph.

2 Kernelization Through Tidying

For a set $\mathcal{F}$ of forbidden induced subgraphs (Fisgs) we outline a general kernelization method for the $\mathcal{F}$-Free Vertex Deletion problem. Here, the task is, given an undirected graph $G = (V, E)$ and an integer $k \geq 0$, to decide whether the graph can be made $\mathcal{F}$-free by deleting at most k vertices. If $\mathcal{F}$ is finite (as we have for s-PCVD, s being a constant), then $\mathcal{F}$-Free Vertex Deletion is clearly fixed-parameter tractable with respect to the parameter k, as directly follows from Cai's [4] general result. Moreover, one may observe that its minimization version is in MIN $\mathrm{F}^{+}\Pi_1$; therefore, using a technique due to Kratsch [10], $\mathcal{F}$-Free Vertex Deletion admits a problem kernel containing $O(k^h)$ vertices, where h is the maximum number of vertices of a Fisg in $\mathcal{F}$. We present an alternative technique to kernelize such vertex deletion problems. While the technique of Kratsch [10] is more general, our approach seems to be useful to obtain smaller problem kernels. For example, the Fisgs for s-PCVD consist of at most $s+1+T_s$ vertices [7], where T_s is the maximum integer satisfying $T_s \cdot (T_s+1) \leq s$; therefore, Kratsch's technique [10] yields an $O(k^{s+1+T_s})$-vertex problem kernel. In contrast, we obtain an $O(k^2s^3)$-vertex problem kernel using a novel method. Our approach comprises the following three main steps.

Approximation Step. This step is to compute an approximate solution X for $\mathcal{F}$-Free Vertex Deletion in polynomial time; let a be the corresponding approximation factor. Obviously, the *residual graph* $G - X$ is $\mathcal{F}$-free. Since X is a factor-a approximate solution, we can abort with returning "no-instance"

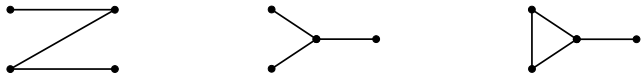


Fig. 1: Minimal forbidden induced subgraphs (FISGs) for 2-plex cluster graphs.

if $|X| > ak$. Hence, otherwise, we proceed knowing that $|X| \leq ak$. It remains to bound the number of vertices in the residual graph $G - X$. To this end, we use the property that $G - X$ is $\mathcal{F}$-free.

This property can be difficult to exploit; however, it is always possible to efficiently find a small vertex set $T \subseteq V \setminus X$ (called *tidying set*) such that in $G-T$ (called the *tidy subgraph*), for each $v \in X$, deleting all vertices in $X \setminus \{v\}$ results in an $\mathcal{F}$-free graph (called the *local tidiness property of* $G - T$). As we will see, this additional property helps in finding suitable data reduction rules to shrink the size of $G - X$. The subsequent *Tidying Step* finds such a tidying set T.

Tidying Step. In this step, polynomial-time data reduction is employed and the tidying set T is computed. Roughly speaking, the data reduction ensures that the tidying set does not become too big.

First, we describe the data reduction. Compute for each $v \in X$ a maximal set $\mathcal{F}(v)$ of FISGs that pairwise intersect exactly in v. If $|\mathcal{F}(v)| > k$, then a "high-degree data reduction rule" deletes v from both G and X and decreases the parameter k by one. The correctness of this rule is easy to verify.

Second, we describe how to compute the tidying set T, show that its size is bounded, and argue that $G-T$ fulfills the local tidiness property. We define the *tidying set* of a vertex $v \in X$ as $T(v) := \bigcup_{F \in \mathcal{F}(v)} V(F) \setminus X$ (that is, all vertices in $V \setminus X$ that are in a FISG of $\mathcal{F}(v)$). The *tidying set* of the whole graph is defined as $T := \bigcup_{v \in X} T(v)$. Since after the high-degree data reduction rule $|\mathcal{F}(v)| \leq k$, we know that $|T(v)| \leq hk$, where h is the maximum number of vertices of a FISG in $\mathcal{F}$; hence, $|T| \leq hak^2$. The local tidiness property of $G - T$ follows directly from the maximality of $\mathcal{F}(v)$. The local tidiness property can be exploited in the final *Shrinking Step*.

Shrinking Step. This is the most unspecified step, which has to be developed using specific properties of the studied vertex deletion problem. In this step, the task is to shrink the tidy subgraph $G - T$ using problem-specific data reduction rules that exploit the local tidiness property. Depending on the strength of these data reduction rules, the total problem kernel size is as follows: as we have seen, the factor-a approximate solution X has size at most ak. The tidying set T based on size-at-most-h FISGs has size at most hak^2. If we shrink the tidy subgraph $G - T$ to at most $f(k)$ vertices, then we obtain a problem kernel with $O(ak + hak^2 + f(k))$ vertices.

In the next section, we present a case study of kernelization through tidying using 2-PCVD. After that, we generalize the approach to s-PCVD with $s > 2$ (Section 4). Finally, we show how to significantly speed up the kernelization.

3 A Problem Kernel for 2-Plex Cluster Vertex Deletion

We specify the steps outlined in Section 2 to obtain a problem kernel for 2-PCVD.

Approximation Step. Following the tidying kernelization method, greedily compute a factor-4 approximate 2-plex cluster graph vertex deletion set (2-*pvd set*) X using the FISG characterization of 2-plex cluster graphs [7] (simply find one of the three four-vertex FISGs (see Figure 1), add its vertices to X, and remove its vertices from the graph, until the remaining graph is a 2-plex cluster graph). If $|X| > 4k$, then simply return "no-instance". Therefore, in the following one may assume that $|X| \leq 4k$. It remains to bound the number of vertices in the residual graph $G - X$.

Using the linear-time algorithm by Guo et al. [7] to find a FISG for 2-plex cluster graphs, we can compute X in $O(k \cdot (n+m))$ time, by either applying the linear-time algorithm at most $k+1$ times or returning "no-instance".

Tidying Step. Let X be the factor-4 approximate 2-pvd set computed in the Approximation Step. For each $v \in X$, greedily compute a maximal set $\mathcal{F}(v)$ of FISGs pairwise intersecting exactly in v. Since the FISGs for 2-PCVD contain four vertices, this can be done in $O(n^3)$ time for each $v \in X$ and therefore in $O(|X| \cdot n^3) = O(k \cdot n^3)$ time in total.[1]

Reduction Rule 1. *If there exists a vertex $v \in X$ such that $|\mathcal{F}(v)| > k$, then delete v from G and X and decrement k by one.*

Lemma 1. *Rule 1 is correct and can be exhaustively applied in $O(n+m)$ time.*

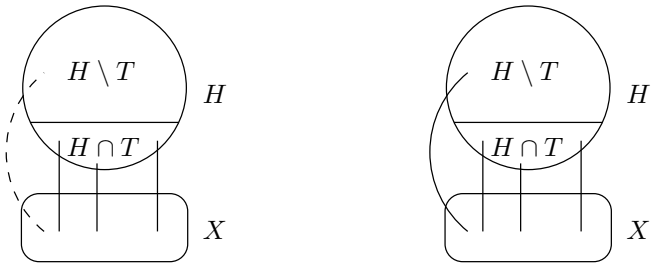
Additionally, we apply a simple and obviously correct data reduction rule in $O(n+m)$ time:

Reduction Rule 2. *Delete connected components from G that form 2-plexes.*

In the following, we assume that G is reduced with respect to Rules 1 and 2. Moreover, X is a 2-pvd set of size at most $4k$, and a maximal set of FISGs $\mathcal{F}(v)$ that pairwise intersect exactly in v shall be computed for each $v \in X$. The tidying set is $T(v) := \bigcup_{F \in \mathcal{F}(v)} V(F) \setminus X$ for each $v \in X$ and the tidying set for the whole graph is $T := \bigcup_{v \in X} T(v)$. Since $|\mathcal{F}(v)| \leq k$, $|X| \leq 4k$, and the FISGs have at most four vertices, we can conclude that $|T| \leq 12k^2$. It remains to bound the number of vertices in $G - (X \cup T)$; more specifically, we bound the number of vertices in the tidy subgraph $G - T$.

Shrinking Step. Since the set $\mathcal{F}(v)$ of FISGs is maximal, the tidy subgraph $G - T$ fulfills the local tidiness property, that is, for each $v \in X$, deleting $X \setminus \{v\}$ from $G - T$ results in an $\mathcal{F}$-free graph.

[1] In Section 4, we present a slightly modified version of our kernelization algorithm that can be performed in $O(kn^2)$ time.



(a) An X-separated set $H \in \mathcal{H}(X)$ (b) A non-X-separated set $H \in \mathcal{H}(X)$

Fig. 2: Diagrams illustrating an X-separated and a non-X-separated set $H \in \mathcal{H}(X)$. The solid lines between X and H denote possible edges, and the dashed line between X and $H \setminus T$ illustrates that there is no edge between these two sets.

Definition 1. *Let $\{G_1, \ldots, G_l\}$ be the set of connected components of $G-X$ and let $\mathcal{H}(X) := \{V(G_1), \ldots, V(G_l)\}$ be the collection of vertex sets of the connected components of $G - X$.*

Since X is a 2-pvd set, each element of $\mathcal{H}(X)$ induces a 2-plex in G.

The local tidiness property helps in finding useful structural information; in particular, observe that each vertex $v \in X$ can be adjacent to vertices of arbitrarily many clusters in $G-X$, but in the tidy subgraph $G-T$, each vertex $v \in X$ is adjacent to vertices of at most two clusters in $G - (T \cup X)$; otherwise, if a vertex $v \in X$ were adjacent to at least three clusters in $G - (T \cup X)$, then there would be a Fisg (more precisely, a $K_{1,3}$) that contains v and three vertices from $G - (T \cup X)$, contradicting the local tidiness property of $G - T$. With such kind of observations, we can show that the local tidiness property implies the following two properties for each $v \in X$. These properties will later be exploited by the data reduction rules and the corresponding proof of the kernel size:

Property 1: There are at most two sets $H \in \mathcal{H}(X)$ with $H \setminus T$ adjacent to v.

Property 2: If there is a set $H \in \mathcal{H}(X)$ such that $H \setminus T$ is adjacent to v, then v is nonadjacent to at most one vertex in $H \setminus T$.

Lemma 2. *The local tidiness property of $G - T$ implies Properties 1 and 2.*

Recall that $|X| \leq 4k$ and that $|T| \leq 12k^2$; it remains to reduce the size of the tidy subgraph $G - T$. To this end, we distinguish between two types of sets in $\mathcal{H}(X)$, namely X-separated and non-X-separated sets:

Definition 2. *A vertex set $H \in \mathcal{H}(X)$ is* X-separated *if $H \setminus T$ is nonadjacent to X. A connected component of $G - X$ is* X-separated *if its vertex set is X-separated.*

See Figure 2 for an illustration. The remainder of this section is mainly devoted to data reduction rules that shrink the size of large sets in $\mathcal{H}(X)$. We deal with X-separated sets and non-X-separated sets separately. The intuitive idea to bound the number of vertices in X-separated sets is as follows. We use the fact that $T \cap H$ is a separator to show that if there are significantly more vertices in $H \setminus T$ than in $H \cap T$, then some vertices in $H \setminus T$ can be deleted from the graph. Together with the size bound for T, one can then obtain a bound on the number of vertices in X-separated sets. The intuitive idea to bound the number of vertices in non-X-separated sets is to use Properties 1 and 2. Roughly speaking, Property 1 guarantees that in $G - T$, each vertex from X is adjacent to at most two sets from $\mathcal{H}(X)$; if there is a large non-X-separated set $H \in \mathcal{H}(X)$, then most of its vertices must have the same neighbors in X due to Property 2. This observation can be used to show that some vertices in $H \setminus T$ can be deleted from the graph.

In the following, we exhibit the corresponding technical details. First, we shrink the size of the X-separated sets.

Bounding the size of X-separated connected components. The following data reduction rule decreases the number of vertices in large X-separated sets in $\mathcal{H}(X)$. Recall that G is reduced with respect to Rules 1 and 2.

Reduction Rule 3. *If there exists a vertex set $H \in \mathcal{H}(X)$ that is X-separated by T such that $|H \setminus T| > |H \cap T| + 1$, then choose an arbitrary vertex from $H \setminus T$ and delete it from G.*

Lemma 3. *Rule 3 is correct and can be exhaustively applied in $O(n + m)$ time.*

Proof. First, we show the correctness. Let $u \in H$ be the vertex that is deleted by Rule 3. If (G, k) is a yes-instance, then obviously $(G - \{u\}, k)$ is a yes-instance as well. Now suppose that $(G - \{u\}, k)$ is a yes-instance. Let S be a 2-pvd set of size at most k for $G - \{u\}$. If S is a 2-pvd set for G, then (G, k) is obviously a yes-instance. Otherwise, u must be contained in a Fisg F in $G - S$; we show that in this case one can use S to construct a 2-pvd set S' of size at most k for G. Since H induces a 2-plex and since H is X-separated, F must contain a vertex $v \in X$ and a vertex $w \in H \cap T$. By the preconditions of Rule 3 and because $|H \cap T| \geq 1$, we have $|H \setminus \{u\}| = |H \setminus (T \cup \{u\})| + |H \cap T| \geq 3$. Thus, all vertices in $H \setminus \{u\}$ are connected to v and the at least $|H \cap T| + 1$ vertices in $H \setminus (T \cup \{u\})$ are nonadjacent to v, because H is X-separated. Since $v, w \notin S$, the 2-pvd set S for $G - \{u\}$ must contain all but at most one vertex from $H \setminus (T \cup \{u\})$, thus $|S \cap (H \setminus T)| \geq |H \cap T|$. Hence $S' := (S \setminus (H \setminus T)) \cup (H \cap T)$ is a 2-pvd set for $G - \{u\}$ of size at most $|S| \leq k$. Since $(H \cap T) \subseteq S'$, $u \in H \setminus T$, and H is X-separated, it follows that S' is also a 2-pvd set of size at most k for G, thus (G, k) is a yes-instance.

For the running time, consider that we can construct $\mathcal{H}(X)$ in $O(n + m)$ time. If for a $H \in \mathcal{H}(X)$ we find that all vertices in $H \setminus T$ are not adjacent to X, then apply Rule 3 to H until $|H \setminus T| \leq |H \cap T| + 1$; deleting vertices from a graph takes $O(n + m)$ time in total. □

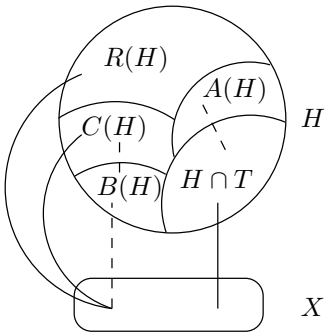


Fig. 3: Illustration of the sets $A(H)$, $B(H)$, $C(H)$, and $R(H)$. Solid lines indicate edges and dashed lines "non-edges". Note that the sets $A(H)$, $B(H)$, $C(H)$, and $H \cap T$ might have pairwise non-empty intersections (which is not relevant for our arguments); to keep the figure simple, they are drawn without intersections.

With Rule 3, the number of vertices in $H \setminus T$ is bounded from above by $|H \cap T|+1$ for each X-separated set $H \in \mathcal{H}(X)$. It remains to shrink the size of non-X-separated sets.

Bounding the size of non-X-separated connected components. Our goal is to find vertices in a non-X-separated set $H \in \mathcal{H}(X)$ that can safely be deleted by a data reduction rule. To this end, we need the following definitions. We call a vertex set $Z \subseteq V$ an *X-module* if any two vertices $u, v \in Z$ satisfy $N(u) \cap X = N(v) \cap X$. We define the following set of "candidate" vertices for deletion.

Definition 3. *Let $H \in \mathcal{H}(X)$. Then, a* redundant subset *$R \subseteq H$ is an X-module in which every vertex $u \in R$ is adjacent to every vertex in $H \setminus R$.*

With this definition, one can state the following data reduction rule; we describe later how a redundant subset of H can be computed.

Reduction Rule 4. *Let $R \subseteq H$ be a redundant subset of some $H \in \mathcal{H}(X)$. If $|R| > k + 3$, then choose an arbitrary vertex from R and delete it from G.*

Lemma 4. *Rule 4 is correct.*

We next show how to efficiently find a redundant subset $R \subseteq H$. See Figure 3 for an illustration of the following definitions. Let $A(H)$ be the set of vertices in H that are nonadjacent to at least one vertex in $H \cap T$; let $B(H)$ be the set of vertices in H that are nonadjacent to at least one vertex from X that has some neighbor in $H \setminus T$; finally, let $C(H)$ be the set of vertices in H that are nonadjacent to some vertex in $B(H)$. Let $\bar{R}(H) := A(H) \cup B(H) \cup C(H)$ and $R(H) := H \setminus (\bar{R}(H) \cup T)$. Intuitively, $\bar{R}(H)$ contains vertices in H that violate Definition 3, guaranteeing that $R(H)$ contains vertices that satisfy Definition 3.

Lemma 5. *For each $H \in \mathcal{H}(X)$, the set $R(H) \subseteq H$ is redundant and the set $\bar{R}(H)$ contains at most $|H \cap T| + 2|N(H \setminus T) \cap X|$ vertices.*

Proof. To prove that $R(H)$ is redundant, one has to show that $R(H)$ is an X-module (that is, $N(u) \cap X = N(v) \cap X$ for all $u, v \in R(H)$) and that each vertex in $R(H)$ is adjacent to all vertices in $H \setminus R(H)$ (see Definition 3). We first show $N(u) \cap X = N(v) \cap X$ for all $u, v \in R(H)$. Assume that $w \in N(u) \cap X$. This implies $w \in N(H \setminus T) \cap X$. If $w \notin N(v) \cap X$, then $v \in B(H)$, contradicting $v \in R(H)$; as a consequence, $N(u) \cap X \subseteq N(v) \cap X$. The inclusion $N(v) \cap X \subseteq N(u) \cap X$ can be shown analogously.

We now show that each vertex in $R(H)$ is adjacent to all vertices in $H \setminus R(H)$. Every vertex in $R(H)$ is (by definition) adjacent to all vertices in $H \cap T$ and $B(H)$. Because each vertex in $A(H) \cup C(H)$ is nonadjacent to a vertex in $B(H) \cup (H \cap T)$ and because H induces a 2-plex, each vertex in $R(H)$ is adjacent to all vertices in $A(H) \cup C(H)$. Thus, each vertex in $R(H)$ is adjacent to every vertex in $H \setminus R(H)$.

Because H induces a 2-plex, $|A(H)| \leq |H \cap T|$ and $|C(H)| \leq |B(H)|$. For each vertex $v \in X$, Property 2 states that there is at most one vertex in $H \setminus T$ that is nonadjacent to v. Thus, $|B(H)| \leq |N(H \setminus T) \cap X|$. As a consequence, $|\bar{R}(H)| = |A(H) \cup B(H) \cup C(H)| \leq |H \cap T| + 2|N(H \setminus T) \cap X|$. □

Lemma 6. *For all $H \in \mathcal{H}(X)$, the set $R(H)$ can be computed in $O(n^2)$ time.*

By Lemma 6, we can compute in $O(n^2)$ time the sets $R(H)$ and shrink them using Rule 4 so that $|R(H)| \leq k + 3$. The number of vertices in $H \setminus (T \cup R(H))$ is upper-bounded by $|H \cap T| + 2|N(H \setminus T) \cap X|$ due to Lemma 5. This shows the following proposition:

Proposition 1. *The exhaustive application of Rule 4 takes $O(n^2)$ time. After that, for each non-X-separated set $H \in \mathcal{H}(X)$, $|H \setminus T|$ has size at most $|H \cap T| + 2|N(H \setminus T) \cap X| + k + 3$.*

Now we have all ingredients to show our main result.

Theorem 1. 2-Plex Cluster Vertex Deletion *admits a problem kernel of at most $52k^2 + 32k$ vertices, which can be computed in $O(k \cdot n^3)$ time.*

Using a more intricate analysis, without introducing further data reduction rules, we can improve the upper bound from Theorem 1 to $40k^2 + 24k$ vertices [3].

4 Generalization to *s*-Plex Cluster Vertex Deletion

In this section, we generalize the kernelization approach for 2-PCVD (Section 3) to s-PCVD with $s > 2$, focusing the presentation on the main differences. In addition, in the last part of this section, we discuss how to speed up the kernelization algorithm.

As in Section 3, we start with the Approximation Step. To greedily compute an approximate solution X, we employ the algorithm by Guo et al. [7], which

finds a T_s-vertex FISG if the given graph contains one; here, T_s is the largest integer satisfying $T_s \cdot (T_s + 1) \leq s$. If X contains more than $k \cdot (s+1+T_s)$ vertices, then return "no-instance". Therefore, in the following, assume that $|X| \in O(sk)$.

For the Tidying Step, for each $v \in X$, compute a maximal set $\mathcal{F}(v)$ of FISGs that pairwise intersect exactly in v. A simple algorithm (trying all subgraphs of $G - \{v\}$ with at most $s + T_s$ vertices) takes $O(|X| \cdot sn^{s+T_s}) = O(s^2k \cdot n^{s+T_s})$ time in total (that is, polynomial time). Then, apply Rule 1, that is, if there exists a vertex $v \in X$ such that $|\mathcal{F}(v)| > k$, then delete v from G and X and decrement k by one. After that, apply a data reduction rule that deletes connected components from G that are s-plexes (cf. Rule 2). The tidying set is $T(v) := \bigcup_{F \in \mathcal{F}(v)} V(F) \setminus X$ for each $v \in X$ and the tidying set for the whole graph is $T := \bigcup_{v \in X} T(v)$. Since $|\mathcal{F}(v)| \leq k$, $|X| = O(sk)$, and the FISGs have at most $O(s)$ vertices, it follows that $|T| = O(s^2k^2)$. It remains to describe the Shrinking Step, which decreases the number of vertices in the tidy subgraph $G - T$. For $G = (V, E)$, let $\mathcal{H}(X) := \{H \subseteq V \mid H$ induces a connected component in $G - X\}$. Analogous to the case $s = 2$, we can show that the local tidiness property implies the following two properties for each $v \in X$:

Property 1: There are at most s sets $H \in \mathcal{H}(X)$ with $H \setminus T$ adjacent to v.

Property 2: If there is a set $H \in \mathcal{H}(X)$ such that $H \setminus T$ is adjacent to v, then v is nonadjacent to at most $2s - 3$ vertices in $H \setminus T$.

We again distinguish between X-separated sets $H \in \mathcal{H}(X)$ and non-X-separated sets $H \in \mathcal{H}(X)$. The main difference compared to the case $s = 2$ is that the proof of the upper bound on the size of the non-X-separated sets is technically more demanding.

Reduction Rule 5 (Generalization of Rule 3). *If there exists a vertex set $H \in \mathcal{H}(X)$ that is X-separated by T such that $|H \setminus T| \geq |H \cap T| + 2s - 2$, then choose an arbitrary vertex from $H \setminus T$ and delete it from G.*

Next, we deal with non-X-separated sets in $\mathcal{H}(X)$. To this end, we need the following definition.

Definition 4 (Generalization of Definition 3). *Let $H \in \mathcal{H}(X)$. Then, we call a subset $R \subseteq H$* redundant *if there is an X-module Z with $R \subseteq Z \subseteq H$ that contains all vertices from H that are nonadjacent to a vertex in R.*

The main difference to Definition 3 for $s = 2$ is that one cannot guarantee that each vertex in a redundant subset R is adjacent to all vertices in $H \setminus R$.

Reduction Rule 6 (Generalization of Rule 4). *Let $R \subseteq H$ be a redundant subset of some $H \in \mathcal{H}(X)$. If $|R| \geq k + 2s$, then choose an arbitrary vertex from R and delete it from G.*

Lemma 7. *Rule 6 is correct.*

To find a redundant subset $R(H) \subseteq H$, one can use the same definitions of $R(H)$ and $\bar{R}(H)$ as for the case $s = 2$; however, the proof that $R(H)$ is a redundant subset becomes slightly more involved.

Lemma 8 (Generalization of Lemma 5). *For each $H \in \mathcal{H}(X)$, the set $R(H) \subseteq H$ is redundant and the set $\bar{R}(H)$ contains $O(s \cdot |H \cap T| + s^2 \cdot |N(H \setminus T) \cap X|)$ vertices.*

Now, we are ready to prove the problem kernel.

Theorem 2. s-PCVD *admits a problem kernel of $O(k^2 s^3)$ vertices, which can be computed in $O(s^2 k \cdot n^{s+T_s})$ time, where T_s is the maximum integer satisfying $T_s \cdot (T_s + 1) \leq s$.*

Speeding up the Kernelization Algorithms. So far, we focused on the kernel size rather than the running time of the kernelization. The bottleneck of the kernelization result given by Theorem 2 is the running time of the simple algorithm that finds for each $v \in X$ a maximal set $\mathcal{F}(v)$ of FISGs that pairwise intersect exactly in v. The maximality of $\mathcal{F}(v)$ was used to prove that the tidying set T fulfills Properties 1 and 2 (of Section 4). We obtain a fast kernelization if we do not demand that $\mathcal{F}(v)$ is maximal; rather, we show that we can compute a set $T(v)$ of bounded size such that Properties 1 and 2 are still fulfilled.

Lemma 9. *Let X be an s-pvd set. Then, for all $v \in X$, a vertex set $T(v)$ with $|T(v)| \leq 2sk$ and satisfying the following properties can be found in $O(|X| \cdot n^2)$ time:*

1. *For each vertex $v \in X$, there are at most s sets $H \in \mathcal{H}(X)$ such that $H \setminus T(v)$ is adjacent to v.*
2. *If there is a vertex $v \in X$ and a set $H \in \mathcal{H}(X)$ such that $H \setminus T(v)$ is adjacent to v, then v is nonadjacent to at most $2s - 3$ vertices in $H \setminus T(v)$.*

Using Lemma 9 instead of the expensive computation of $T(v)$ in Section 4 and a few additional tricks [3], the following result can be shown.

Theorem 3. s-PCVD *admits a problem kernel of $O(k^2 s^3)$ vertices, which can be computed in $O(ksn^2)$ time.*

5 Conclusion

Our results are based on linking kernelization with polynomial-time approximation, dealing with vertex deletion problems whose goal graphs are characterized by forbidden induced subgraphs. This is a rich class of graphs, among others containing various cluster graphs. When applicable, our method may allow for significantly smaller problem kernel sizes than the more general method by Kratsch [10].

As to future work, it would be desirable to start a general study under which conditions fixed-parameter tractable vertex deletion problems possess polynomial-size kernels.

References

1. F. N. Abu-Khzam. A kernelization algorithm for d-Hitting Set. *J. Comput. System Sci.*, 2009. Available electronically.
2. B. Balasundaram, S. Butenko, and I. V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Oper. Res.*, 2009. Avaiable electronically.
3. R. van Bevern. A quadratic-vertex problem kernel for s-plex cluster vertex deletion. Studienarbeit, Friedrich-Schiller-Universität Jena, Germany, 2009.
4. L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996.
5. E. J. Chesler, L. Lu, S. Shou, Y. Qu, J. Gu, J. Wang, H. C. Hsu, J. D. Mountz, N. E. Baldwin, M. A. Langston, D. W. Threadgill, K. F. Manly, and R. W. Williams. Complex trait analysis of gene expression uncovers polygenic and pleiotropic networks that modulate nervous system function. *Nat. Genet.*, 37(3):233–242, 2005.
6. V. J. Cook, S. J. Sun, J. Tapia, S. Q. Muth, D. F. Argüello, B. L. Lewis, R. B. Rothenberg, P. D. McElroy, and the Network Analysis Project Team. Transmission network analysis in tuberculosis contact investigations. *J. Infect. Dis.*, 196:1517–1527, 2007.
7. J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. A more relaxed model for graph-based data clustering: s-plex editing. In *Proc. 5th AAIM*, volume 5564 of *LNCS*, pages 226–239. Springer, 2009.
8. J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007.
9. F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.*, 2009. Available electronically.
10. S. Kratsch. Polynomial kernelizations for MIN $\mathrm{F}^{+}\Pi_1$ and MAX NP. In *Proc. 26th STACS*, pages 601–612. IBFI Dagstuhl, Germany, 2009.
11. D. Marx and I. Schlotter. Parameterized graph cleaning problems. *Discrete Appl. Math.*, 2009. Available electronically.
12. N. Memon, K. C. Kristoffersen, D. L. Hicks, and H. L. Larsen. Detecting critical regions in covert networks: A case study of 9/11 terrorists network. In *Proc. 2nd ARES*, pages 861–870. IEEE Computer Society Press, 2007.
13. S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *J. Math. Sociol.*, 6:139–154, 1978.
14. R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Appl. Math.*, 144(1–2):173–182, 2004.