Preprint. Article published in Algorithmica.
DOI 10.1007/s00453-011-9492-7

Approximation and Tidying—A Problem Kernel for s-Plex Cluster Vertex Deletion

René van Bevern · **Hannes Moser** · **Rolf Niedermeier**

Submitted: May 28, 2010 / Accepted: Jan 17, 2011

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 based on "approximation and tidying" for kernelizing vertex deletion problems whose goal graphs can be characterized by forbidden induced subgraphs. The method exploits polynomial-time approximation results and thus provides a useful link between approximation and kernelization. Employing "approximation and tidying", we develop data reduction rules that, in $O(ksn^2)$ time, transform an s-PLEX CLUSTER VERTEX DELETION instance with n vertices into an equivalent instance with $O(k^2s^3)$ vertices, yielding a problem kernel. To this end, we also show how to exploit structural properties of the specific problem in order to significantly improve the running time of the proposed kernelization method.

Keywords computational intractability · NP-hard graph problem · polynomial-time data reduction · fixed-parameter tractability · graph-based data clustering

Supported by the DFG, project AREG, NI 369/9. An extended abstract of this paper appeared under the title "Kernelization Through Tidying—A Case-Study Based on s-Plex Cluster Vertex Deletion" in *Proceedings of the 9th Latin American Theoretical Informatics Symposium (LATIN'10)*, Oaxaca, México, April 2010, volume 6034 of Lecture Notes in Computer Science, pages 528–539, Springer, 2010. Whereas in the extended abstract the proofs focused on 2-PLEX CLUSTER VERTEX DELETION, here we present the kernelization for the general s-PLEX CLUSTER VERTEX DELETION problem, which is only slightly more technical.

René van Bevern and Rolf Niedermeier
Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, D-10623 Berlin, Germany,
E-mail: rene.vanbevern@tu-berlin.de, rolf.niedermeier@tu-berlin.de

Hannes Moser
Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany, E-mail: hannes.moser@uni-jena.de

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 the introduced kernelization through "approximation and tidying", we provide polynomial-time data reduction rules that transform, in $O(ksn^2)$ time, an s-PLEX CLUSTER VERTEX DELETION instance into an equivalent instance comprising $O(k^2s^3)$ vertices. Here, it is essential that the number of vertices in the output instance only depends on the *parameter* k, not on the size of the input graph. The data reduction rules yield a so-called *problem kernel* [4, 13]. In the following, we define and motivate the problem s-PLEX CLUSTER VERTEX DELETION and discuss the main contributions of this work.

Motivation Many vertex deletion problems in graphs can be considered as "graph cleaning procedures", see Diaz and Thilikos [8] or Marx and Schlotter [21]. This view particularly applies to graph-based data clustering, where the graph vertices represent data items and there is an edge between two vertices if and only if the corresponding items are similar enough [28, 30]. The task is then to find *clusters* of the given items such that items within a cluster are similar, while items from different clusters are dissimilar. The resulting clusters may reveal information about the input data. For example, consider a group of diseased people. From this group, we can obtain a graph whose vertices represent the persons and where an edge is drawn between two persons with similar symptoms. A clustering of the so-obtained graph might reveal that the group of diseased people actually consists of multiple people subgroups (the clusters of the graph), each suffering from a distinct disease that must be cured independently. Ideally, the graph to be clustered has only cliques as connected components: then, by considering each connected component as a cluster, clearly every two items in the same cluster are similar, whereas items in different clusters are dissimilar. We call a graph that has only cliques as connected components a *cluster graph*. However, 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 [30]. A recent example for this is the NP-hard CLUSTER VERTEX DELETION problem [17], where the task is to delete as few vertices as possible such that the resulting graph is a cluster graph. The vertices removed might be considered as outliers or might be identified as "hubs": people having and spreading multiple diseases at once. The drawback of the CLUSTER VERTEX DELETION approach is that asking the connected components of a graph to be cliques often seems overly restrictive [6, 29]. As a result, too many vertex deletions from the input graph may be necessary to transform it into a cluster graph, too strongly changing the original data. Therefore, we relax the requirement of each connected component being a clique and allow (by varying a parameter s) to balance the number of vertices to be deleted against the density and size of the resulting clusters: in the NP-hard [20] s-PLEX CLUSTER VERTEX DELETION problem studied here, we replace cliques by so-called s-plexes (where s is typically a small constant):

> s-PLEX CLUSTER VERTEX DELETION
> **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 [29]. Subsequently, we refer to the solution set S as *s-plex cluster vertex deletion set* and call a disjoint union of s-plexes an *s-plex cluster graph*. The concept of s-plexes has recently received considerable interest in various fields, see, e.g., [3, 7, 15, 22, 23, 25, 31]. Summarizing, s-PLEX CLUSTER VERTEX DELETION blends and extends previous studies on CLUSTER VERTEX DELETION [17] (which is the same as 1-PLEX CLUSTER VERTEX DELETION) and on s-PLEX CLUSTER EDITING [15], where in the latter problem one tries to add and delete at most k *edges* to transform a graph into an s-plex cluster graph.

Previous work The two "sister problems" of s-PLEX CLUSTER VERTEX DELETION, namely s-PLEX CLUSTER EDITING and CLUSTER VERTEX DELETION, have been subject to recent research [15, 17]. For CLUSTER VERTEX DELETION, Hüffner et al. [17] have developed fixed-parameter algorithms using the iterative compression technique [14] introduced by Reed et al. [27]. Their algorithm solves CLUSTER VERTEX DELETION on a graph with n vertices and m edges in $O(2^k \cdot n^2(m+n\log n))$ time, where k is the number of allowed vertex deletions. In the same work, Hüffner et al. [17] also described how to compute an $O(k^2)$-vertex problem kernel for CLUSTER VERTEX DELETION in $O(k^9+mn)$ time, exploiting a problem kernel for 3-HITTING SET by Abu-Khzam [1]. Roughly speaking, a problem kernel is an equivalent (smaller) instance whose size is bounded by a function only depending on a parameter and computable in polynomial time. Guo et al. [15] have shown a problem kernel with $O(ks^2)$ vertices for s-PLEX CLUSTER EDITING computable in $O(n^4)$ time, where k is the number of allowed edge modifications. In the same work, Guo et al. [15] developed a forbidden induced subgraph characterization for s-plex cluster graphs: every graph that is not an s-plex cluster graph contains a forbidden induced subgraph consisting of at most $s+\sqrt{s}+1$ vertices that can be found in $O(s(n+m))$ time. This yields a trivial search tree algorithm to solve s-PLEX CLUSTER VERTEX DELETION in $O((s+\sqrt{s}+1)^k \cdot s(n+m))$ time by repeatedly searching for a forbidden induced subgraph in the input graph and branching into all possibilities of deleting one of its vertices. A problem kernel for s-PLEX CLUSTER VERTEX DELETION can be obtained from a general result from Kratsch [19], who showed polynomial-size problem kernels for constant-factor approximable problems contained in the classes MIN $\mathrm{F}^+\Pi_1$ and MAX NP. Since s-PLEX CLUSTER VERTEX DELETION is contained in MIN $\mathrm{F}^+\Pi_1$, applying Kratsch's method [19] yields an $O(k^{s+\sqrt{s}+1})$-vertex problem kernel for s-PLEX CLUSTER VERTEX DELETION. His method is based on problem kernels for HITTING SET, which have also been studied by Abu-Khzam [1].

Our results Complementing and extending results for CLUSTER VERTEX DELETION [17], we prove an $O(k^2s^3)$-vertex problem kernel for s-PLEX CLUSTER VERTEX DELETION that can be found in $O(ksn^2)$ time. We emphasize that the corresponding kernelization algorithm for s-PLEX CLUSTER VERTEX DELETION is completely different from the $O(ks^2)$-vertex-kernelization for s-PLEX CLUSTER EDITING [15]. As vertex deletion is a "more powerful" operation than edge modification, a larger problem kernel in the case of s-PLEX CLUSTER VERTEX DELETION does not come unexpectedly. Our main conceptual contribution is the "approximation and tidying" kernelization method outlined in Section 3. A decisive advantage over the $O(k^{s+\sqrt{s}+1})$-vertex problem kernel for s-PLEX CLUSTER VERTEX DELETION that can be obtained from a general

result from Kratsch [19] is that, in our $O(k^2 s^3)$-vertex bound, the value of s does not influence the degree of the polynomial in k.

Organization of this paper Section 2 explains the concept of kernelization and basic notation. This is followed by an introduction of the approximation and tidying method in Section 3. Section 4 applies the method to s-PLEX CLUSTER VERTEX DELETION. Section 5 shows a more efficient execution of the proven data reduction rules. Finally, Section 6 summarizes our findings and provides an outlook on future research challenges.

2 Preliminaries

Parameterized algorithmics [10, 12, 26] is an approach to find optimal solutions for NP-hard problems. The idea is to accept the seemingly inevitable combinatorial explosion, but to confine it to one aspect of the problem, the *parameter*. More precisely, a problem is *fixed-parameter tractable* (FPT) with respect to a parameter k if there is an algorithm solving any problem instance of size n in $f(k) \cdot n^{O(1)}$ time for some computable function f. A common method in parameterized algorithmics is to provide polynomial-time executable *data reduction rules* that lead to a *problem kernel* [4, 13]. This is perhaps the concept in parameterized algorithmics with the widest practical relevance [13, 16].

Problem kernelization Polynomial-time data reduction and problem kernelization is a core tool of parameterized algorithmics [4, 13]. 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 if and only if (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.

Notation We only consider undirected, finite 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 $m := |E|$. We use $V(G)$ and $E(G)$, respectively, to denote the vertex and edge set of a graph G. We call two vertices $v,w \in V$ *adjacent* or *neighbors* if $\{v,w\} \in E$. 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 $N(U) \cap W \neq \emptyset$. A *path* in G from v_1 to v_ℓ is a sequence $(v_1, v_2, \ldots, v_\ell) \in V^\ell$ of vertices with $\{v_i, v_{i+1}\} \in E$ for $i \in \{1, \ldots, \ell-1\}$. We call two vertices v and w *connected* in G if there is 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 isomorphic to a graph in $\mathcal{F}$ as an induced subgraph. For the class of $\mathcal{F}$-free graphs, the graphs in $\mathcal{F}$ are called *forbidden induced subgraphs*.

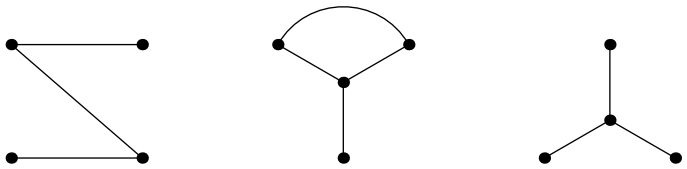


Fig. 1: Minimal forbidden induced subgraphs for 2-plex cluster graphs.

3 The Method: Approximation and Tidying

For a set $\mathscr{F}$ of graphs, we outline a kernelization method for $\mathscr{F}$-FREE VERTEX DELETION: given an undirected graph $G = (V, E)$ and an integer $k \geq 0$, decide whether the graph can be made $\mathscr{F}$-free by deleting at most k vertices. For example, if we let $\mathscr{F}$ contain the graphs shown in Figure 1, then $\mathscr{F}$-FREE VERTEX DELETION is precisely 2-PLEX CLUSTER VERTEX DELETION [15]. For finite sets $\mathscr{F}$, the problem $\mathscr{F}$-FREE VERTEX DELETION is clearly fixed-parameter tractable with respect to the parameter k, obtained by a straightforward branching strategy. Moreover, one may observe that its minimization version is in MIN $\mathrm{F}^{+}\Pi_1$, a class of minimization problems whose optimum solution can be expressed using a certain type of universally quantified first-order logic formulae [18]. Therefore, using a technique due to Kratsch [19], $\mathscr{F}$-FREE VERTEX DELETION admits a problem kernel containing $O(k^h)$ vertices, where h is the maximum number of vertices of a graph in $\mathscr{F}$. We present an alternative technique to kernelize $\mathscr{F}$-FREE VERTEX DELETION problems for finite sets $\mathscr{F}$. While the technique of Kratsch [19] is more general, the approach presented here seems to be useful to obtain smaller problem kernels. For example, the forbidden induced subgraphs for s-PLEX CLUSTER VERTEX DELETION consist of at most $s+\sqrt{s}+1$ vertices [15]; therefore, Kratsch's technique [19] yields an $O(k^{s+\sqrt{s}+1})$-vertex problem kernel. In contrast, we obtain an $O(k^2 s^3)$-vertex problem kernel using the approximation and tidying method described below. It comprises the following three steps.

Approximation Step The task of this step is to compute in polynomial time an approximate solution X for $\mathscr{F}$-FREE VERTEX DELETION. Let c be the corresponding approximation factor. Obviously, the *residual graph* $G-X$ is $\mathscr{F}$-free. Since X is a factor-c approximation, we can abort with returning "no-instance" if $|X| > ck$. Otherwise, we proceed knowing that $|X| \leq ck$. It remains to shrink and bound the number of vertices in the residual graph $G-X$. To this end, we use the property that $G-X$ is $\mathscr{F}$-free. The partition of the input graph G into an $\mathscr{F}$-free graph $G-X$ and a vertex set X is illustrated in Figure 2.

Tidying Step The Approximation Step computes a vertex set X with $|X| \leq ck$ such that $G-X$ is $\mathscr{F}$-free. However, since $\mathscr{F}$-freeness can be difficult to exploit, the task of the Tidying Step is structuring the input further to make problem-specific data reduction rules applicable. To this end, we compute in polynomial time a vertex subset T of $G-X$ (called *tidying set*) such that re-inserting any single vertex $v \in X$ into $G-(T \cup X)$ leaves it $\mathscr{F}$-free. This property is illustrated in Figure 3; we refer to it as *tidiness property* and call $G-T$ the *tidy subgraph* of G. The tidiness property enables us to find a family of induced subgraphs of G, each being $\mathscr{F}$-free and overlapping with both $G-X$ and $G[X]$. We will see that this helps us analyzing the structure of the connections between the

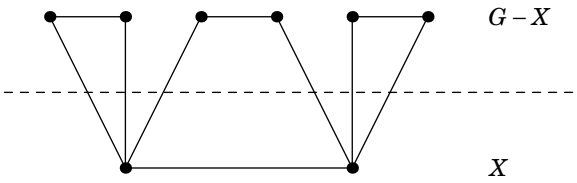


Fig. 2: For the set $\mathscr{F}$ of graphs shown in Figure 1, a partition of the input graph G into an $\mathscr{F}$-free graph (that is, 2-plex cluster graph) $G-X$ and an approximate solution X is shown.

approximate solution X and the residual graph $G-X$. This, in turn, enables us to find suitable data reduction rules to shrink $G-X$.

To compute the tidying set T, greedy algorithms and polynomial-time data reduction are employed. Roughly speaking, the data reduction ensures that the tidying set does not become too large.

First, we describe the data reduction. Compute for each $v \in X$ a maximal set $\mathscr{F}(v)$ of forbidden induced subgraphs that pairwise intersect exactly in v. If $|\mathscr{F}(v)| > k$, then v must be deleted in any attempt to transform the input graph G into an $\mathscr{F}$-free graph by at most k vertex deletions: if v was not deleted, then a vertex of each forbidden induced subgraph in $\mathscr{F}(v)$ would have to be deleted. This, however, is not affordable since $|\mathscr{F}(v)| > k$. Hence, a "high-degree data reduction rule" can delete v from both G and X and decrease the parameter k of allowed vertex deletions by one.

Second, we describe how to compute the tidying set T, show that its size is bounded, and argue that $G-T$ fulfills the tidiness property. We define the *tidying set* of a vertex $v \in X$ as $T(v) := \bigcup_{F \in \mathscr{F}(v)} V(F) \setminus X$ (that is, all vertices in $V \setminus X$ that are in a forbidden induced subgraph of $\mathscr{F}(v)$). 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 $|\mathscr{F}(v)| \leq k$, we know that $|T(v)| \leq ak$, where a is the maximum number of vertices of a forbidden induced subgraph in $\mathscr{F}$. Combining this with $|X| \leq ck$, one gets $|T| \leq ack^2$. The tidiness property of $G-T$ now follows easily: assume that, for some $v \in X$, the graph $G-(T \cup X \setminus \{v\})$ is not $\mathscr{F}$-free. Then, there is a forbidden induced subgraph F in G that contains v and no vertices from T. As this implies that F contains no vertex from $\mathscr{F}(v)$, this contradicts the maximality of $\mathscr{F}(v)$. The tidiness property is 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 tidiness property. From these data reduction rules, one obtains a graph G' comprising vertices from the approximate solution X, from the tidying set T, and from the tidy subgraph $G-T$, each having been shrunken by data reduction rules. These data reduction rules also yield a new parameter value k', which corresponds to the number of allowed vertex deletions in G'. To make sure that all data reduction rules are *correct*, one has to prove that (G',k') is a yes-instance if and only if the input (G,k) is a yes-instance. This, together with a size bound on G', shows that (G',k') is a problem kernel. Depending on the strength of the applied data reduction rules, the resulting problem kernel size is as follows: the factor-c approximate solution X has size at most ck. The tidying set T based on size-at-most-a forbidden induced subgraphs

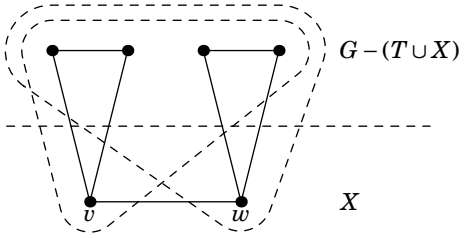


Fig. 3: For $\mathscr{F}$ being the set of graphs shown in Figure 1 and a tidying set T, the tidy subgraph $G-T$ is shown. The tidiness property is satisfied: adding one of the vertices v and w (but not both) to $G-(T\cup X)$ results in an $\mathscr{F}$-free graph, that is, the induced subgraphs surrounded by the dashed lines are 2-plex cluster graphs.

has size at most ack^2. If one shrinks the tidy subgraph $G-T$ to at most $f(k)$ vertices, then one obtains a problem kernel with $ck+ack^2+f(k)$ vertices.

Summarizing, approximation and tidying works by first computing in polynomial time an approximate solution for a vertex deletion problem. Based on the approximate solution, the input graph is further structured (in our words, tidied up) in order to make problem-specific data reduction rules applicable. Herein, the Approximation Step and the Tidying Step work for vertex deletion problems for graph properties characterized by a finite set of forbidden induced subgraphs. One example for such a problem is s-PLEX CLUSTER VERTEX DELETION.

In the next section, we present a problem kernel for s-PLEX CLUSTER VERTEX DELETION using approximation and tidying. On the one hand, we present the problem-specific Shrinking Step. On the other hand, we also show how the problem-independent Approximation Step and Tidying Step can be executed efficiently if one exploits properties of the specific graph property to be established.

4 Kernelization for s-Plex Cluster Vertex Deletion

In this section, we give a detailed description of the steps outlined in Section 3 to obtain a problem kernel for s-PLEX CLUSTER VERTEX DELETION. First, we show how the Approximation Step and the Tidying Step can be executed for s-PLEX CLUSTER VERTEX DELETION. Then, we give a detailed presentation of the Shrinking Step.

4.1 Approximation Step

Following the approximation and tidying kernelization method, we greedily compute a constant-factor approximate s-plex cluster vertex deletion set X using the forbidden induced subgraph characterization of s-plex cluster graphs [15]. To this end, we employ an $O(s\cdot(n+m))$-time algorithm by Guo et al. [15] that finds, in a graph that is not an s-plex cluster graph, a forbidden induced subgraph of at most $c=s+\sqrt{s}+1$ vertices. We repeatedly find a forbidden induced subgraph in the given input graph G, add its vertices to X, and remove its vertices from G, until G is an s-plex cluster graph. In this way, we find pairwise vertex-disjoint forbidden induced subgraphs for s-plex cluster

graphs in G. If more than k such forbidden induced subgraphs are found, then it is clear that (G,k) is a no-instance. Otherwise, we obtain a set $|X| \leq ck \in O(ks)$ such that $G-X$ is an s-plex cluster graph. Hence, the computation of X takes $O(ks \cdot (n+m))$ time, as we apply Guo et al.'s [15] algorithm at most $k+1$ times.

It remains to bound the number of vertices in the residual graph $G-X$. To this end, using the following definition, we make observations about the connected components of $G-X$.

Definition 1 $\mathscr{H}(X)$ is the collection of vertex sets of the connected components of $G-X$.

Since X is an s-plex cluster vertex deletion set, each set in $\mathscr{H}(X)$ induces an s-plex in G.

4.2 Tidying Step

Let X be the approximate s-plex cluster vertex deletion set computed in the Approximation Step. For each $v \in X$, greedily compute a maximal set $\mathscr{F}(v)$ of forbidden induced subgraphs pairwise intersecting exactly in v. Since the forbidden induced subgraphs for s-PLEX CLUSTER VERTEX DELETION contain at most $s+\sqrt{s}+1$ vertices [15], this can be done in $O(n^{s+\sqrt{s}+1})$ time for each $v \in X$ and therefore in $O(|X| \cdot n^{s+\sqrt{s}+1}) = O(ksn^{s+\sqrt{s}+1})$ time in total.[1]

Reduction Rule 1 *If there is a vertex $v \in X$ such that $|\mathscr{F}(v)| > k$, then delete v from G and from X and decrement the number k of allowed vertex deletions by one.*

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

Proof If an s-plex cluster vertex deletion set does not contain v, then it contains one vertex of each of the (more than k) forbidden induced subgraphs in $\mathscr{F}(v)$. Thus, every size-k s-plex cluster vertex deletion set must contain v. For the running time, we use the following simple data structure. Create $k+1$ lists and, for $i \leq k$, add a vertex $v \in X$ to list i if $|\mathscr{F}(v)| = i$; add $v \in X$ to list $k+1$ if $|\mathscr{F}(v)| \geq k+1$. These lists can be created in $O(n+m)$ time. With their help, the deletion of all vertices $v \in X$ with $|\mathscr{F}(v)| > k$ is possible in $O(n+m)$ time while simultaneously decrementing k. □

In Reduction Rule 1, we identify vertices that must be part of any s-plex cluster vertex deletion set of size at most k. Hence, we remove them from the input graph G and search for a s-plex cluster vertex deletion set of size k' in the remaining graph G', where k' is k decremented by the number of vertices removed by Reduction Rule 1. If we find an s-plex cluster vertex deletion set S for G', then we can extend it to an s-plex cluster vertex deletion set for G by adding the vertices to S that we previously identified to be part of any s-plex cluster vertex deletion set of size at most k. The data reduction rules in the remainder of this paper will be substantially different: we remove vertices from the input graph G to obtain a smaller graph G' but do *not* put them into an s-plex cluster vertex deletion set. Instead, we show how any s-plex cluster vertex deletion set of size at most k for G' can be turned into an s-plex cluster vertex

[1] In Section 5, we present a slightly modified version of our kernelization algorithm that can be performed in $O(ksn^2)$ time.

deletion set of size at most k for G. As every s-plex cluster vertex deletion set for G is also an s-plex cluster vertex deletion set for G', this shows that (G,k) is a yes-instance if and only if (G',k) is a yes-instance, thus establishing the correctness of our data reduction rules. The first data reduction rule of this kind is the following:

Reduction Rule 2 *If there is a vertex set $H \in \mathscr{H}(X)$ being nonadjacent to X, then remove it from G.*

Lemma 2 *Reduction Rule 2 is correct and can be exhaustively applied in $O(n+m)$ time.*

Proof This data reduction rule is obviously correct, as each set $H \in \mathscr{H}(X)$ that is nonadjacent to X induces an isolated s-plex in G. For the running time, observe that $\mathscr{H}(X)$ can be constructed in $O(n+m)$ time: using breadth-first search, we remember for each vertex of $G-X$ to which connected component of $G-X$ it belongs and put it into a set in $\mathscr{H}(X)$ accordingly. It remains to remove sets in $\mathscr{H}(X)$ from G that have no neighbors in X. □

In the following, we assume that the input graph G is reduced with respect to Reduction Rules 1 and 2. Moreover, we assume that X is an s-plex cluster vertex deletion set of size $O(ks)$, and that a maximal set of forbidden induced subgraphs $\mathscr{F}(v)$ that pairwise intersect exactly in v has been computed for each $v \in X$. The tidying set is $T(v) := \bigcup_{F \in \mathscr{F}(v)} V(F) \setminus X$ for each $v \in X$ and the tidying set for the whole graph is $T := \bigcup_{v \in X} T(v)$. Since the set $\mathscr{F}(v)$ of forbidden induced subgraphs is maximal, the tidy subgraph $G-T$ fulfills the tidiness property, that is, for each $v \in X$, deleting $X \setminus \{v\}$ from $G-T$ results in an $\mathscr{F}$-free graph. Moreover, since each forbidden induced subgraph in $\mathscr{F}(v)$ has at most $s+\sqrt{s}+1 \in O(s)$ vertices, we conclude $|T| \in O(k^2 s^2)$ from $|\mathscr{F}(v)| \leq k$ and from $|X| \in O(ks)$.

So far, we have bounded $|T \cup X|$. It remains to shrink and bound the number of vertices in $G-(T \cup X)$; more specifically, we bound the number of vertices in the tidy subgraph $G-T$. To this end, we show a way to execute the Shrinking Step for s-Plex Cluster Vertex Deletion, which is the main technical contribution of this work.

4.3 Shrinking Step

We now show how the tidiness property can be exploited in designing further problem-specific data reduction rules. To this end, we need the following lemma:

Lemma 3 *An s-plex containing at least $2s-1$ vertices is a connected graph.*

Proof Let $G=(V,E)$ be an s-plex with more than one connected component. Because G is an s-plex, a vertex in G is nonadjacent to at most $s-1$ other vertices in G. Let $W \subseteq V$ be the vertex set of a connected component of G. Because a vertex in W is nonadjacent to all vertices in $V \setminus W$, we have $|V \setminus W| \leq s-1$ and $|W| \leq s-1$. Therefore, $|V| \leq 2s-2$. □

In the Shrinking Step, it is crucial to exploit the fact that X is an s-plex cluster vertex deletion set, implying that $G-X$ is an s-plex cluster graph. The tidiness property now reveals useful structural information about the connections of the s-plex cluster vertex deletion set X to the connected components of $G-X$. For example, observe that each vertex $v \in X$ can be adjacent to vertices of arbitrarily many connected components of $G-X$. In contrast, below we show that in the tidy subgraph $G-T$, each vertex $v \in X$

is adjacent to vertices of at most one "large" connected component of $G-(T\cup X)$. Specifically, we prove the following lemma, which will be exploited by further data reduction rules and the proof of the kernel size:

Lemma 4 *The tidiness property of $G-T$ implies for each $v\in X$:*

1. *There are at most s sets $H\in\mathscr{H}(X)$ such that $H\setminus T$ is adjacent to v.*
2. *If there is more than one set $H\in\mathscr{H}(X)$ such that $H\setminus T$ is adjacent to v, then each such set H satisfies $|H\setminus T|\leq 2s-2$.*
3. *If there is a set $H\in\mathscr{H}(X)$ such that $H\setminus T$ is adjacent to v, then v is nonadjacent to at most $2s-3$ vertices in $H\setminus T$.*

Proof We show that if one of the properties listed in Lemma 4 is not satisfied for some $v\in X$, then there is a forbidden induced subgraph F with $V(F)\cap X=\{v\}$ and $V(F)\cap T=\emptyset$, contradicting the tidiness property of $G-T$.

(1) Assume that there are $s+1$ sets $H_1,\ldots,H_{s+1}$ such that a vertex $v\in X$ is adjacent to $u_i\in H_i\setminus T$ for $i\in\{1,\ldots,s+1\}$. The vertices u_i are connected because they are neighbors of v. Additionally u_1 is nonadjacent to the s vertices $u_2,\ldots,u_{s+1}$. Hence, $G[\{v,u_1,\ldots,u_{s+1}\}]$ is not an s-plex cluster graph and, therefore, a forbidden induced subgraph.

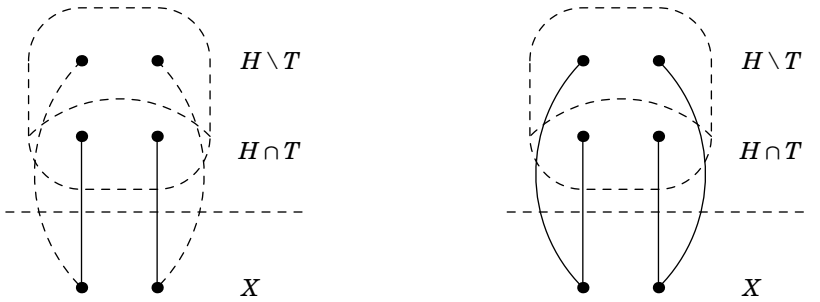
(2) Assume that for a vertex $v\in X$, there are two sets $U,W\in\mathscr{H}(X)$ such that v is adjacent to the vertices $u\in U\setminus T$ and $w\in W\setminus T$. Moreover, assume that $W\setminus T$ contains at least $2s-1$ vertices. Then, since $G[W]$ is an s-plex, $G[W\setminus T]$ is also an s-plex. Moreover, because $G[W\setminus T]$ contains at least $2s-1$ vertices, $G[W\setminus T]$ is connected according to Lemma 3. The vertex $u\in U\setminus T$ is nonadjacent to at least $2s-1$ vertices in $W\setminus T$, but $F:=G[\{u,v\}\cup(W\setminus T)]$ is connected, since u and w are neighbors of v. Hence, F is not an s-plex cluster graph; F is a forbidden induced subgraph.

(3) Assume that there is a set $H\in\mathscr{H}(X)$ such that a vertex $v\in X$ is adjacent to $u\in H\setminus T$ and nonadjacent to a set $U\subseteq H\setminus T$ with $|U|\geq 2s-2$. Because $\{u\}\cup U\subseteq H$ and since $G[H]$ is an s-plex, also $G[\{u\}\cup U]$ is an s-plex. Moreover, $G[\{u\}\cup U]$ contains at least $2s-1$ vertices and, hence, is connected according to Lemma 3. It follows that v, being a neighbor of u, is connected but nonadjacent to the $2s-2$ vertices in U. Thus, $G[\{v,u\}\cup U]$ is a forbidden induced subgraph. □

Toward our goal of obtaining a problem kernel, we have already computed an approximate solution X and the tidying set T with $|T\cup X|\in O(s^2k^2)$ in Section 4.1 and Section 4.2. Therefore, it still remains to reduce the size of $G-(T\cup X)$, that is, to reduce the size of the set $H\setminus T$ for each $H\in\mathscr{H}(X)$. To this end, we distinguish between two types of sets in $\mathscr{H}(X)$:

Definition 2 A vertex set $H\in\mathscr{H}(X)$ is *X-separated* if $H\setminus T$ is nonadjacent to X and *non-X-separated* otherwise.

See Figure 4 for an illustration. The remainder of this section is mainly devoted to data reduction rules that shrink the size of large sets in $\mathscr{H}(X)$. 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 $H\cap T$ 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 removed 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



(a) X-separated set $H \in \mathscr{H}(X)$ (b) non-X-separated set $H \in \mathscr{H}(X)$

Fig. 4: Sets $H \in \mathscr{H}(X)$ are shown partitioned into their parts $H \cap T$ and $H \setminus T$. The solid lines between X and H denote possible edges, and the dashed lines between X and $H \setminus T$ illustrate that there is no edge between these two sets. Here, T is the tidying set.

the properties stated in Lemma 4. Roughly speaking, Lemma 4(1) and Lemma 4(2) guarantee that each vertex from X is adjacent to only one large connected component of $G - T$. From Lemma 4(3), it then follows that if there is a large non-X-separated set $H \in \mathscr{H}(X)$, then most of its vertices have the same neighbors in X. This observation can be used to show that some vertices in $H \setminus T$ can be removed from the graph. In the following, we exhibit the corresponding technical details. First, we shrink the size of the X-separated sets. After that, we shrink the size of the non-X-separated sets.

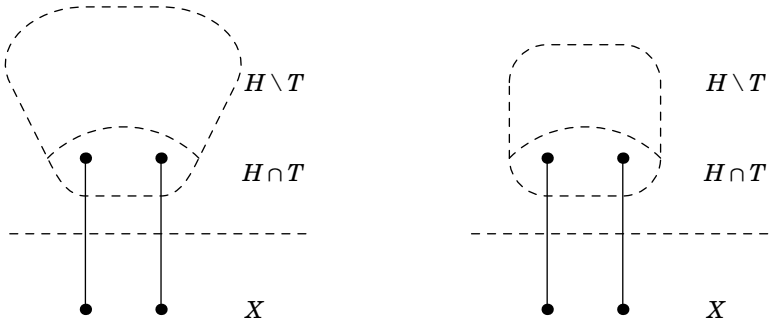
4.3.1 Shrinking X-separated Sets

The following data reduction rule decreases the number of vertices in large X-separated sets. Intuitively, because each X-separated set $H \in \mathscr{H}(X)$ induces an s-plex, any optimal s-plex cluster vertex deletion set will, if $H \setminus T$ is much larger than $H \cap T$, cut off H from X by deleting $H \cap T$ instead of deleting some vertices in $H \setminus T$. Hence, for X-separated sets $H \in \mathscr{H}(X)$ for which $H \setminus T$ is much larger than $H \cap T$, we shrink $H \setminus T$ such that an optimal s-plex cluster vertex deletion set will still delete $H \cap T$. This is illustrated in Figure 5.

Reduction Rule 3 *If there is a vertex set $H \in \mathscr{H}(X)$ being X-separated by T such that $|H \setminus T| > |H \cap T| + 2s - 3$, then choose an arbitrary vertex from $H \setminus T$ and remove it from G.*

Lemma 5 *Reduction 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 removed by Reduction 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 an s-plex cluster vertex deletion set of size at most k for $G-\{u\}$. If S is an s-plex cluster vertex deletion set for G, then (G,k) is obviously a yes-instance. Otherwise, u is contained in a forbidden induced subgraph F in $G-S$; we show that in this case one can use S to construct



(a) before data reduction (b) after data reduction

Fig. 5: Illustration of Reduction Rule 3. Intuitively, if for any X-separated set $H \in \mathscr{H}(X)$, the set $H \setminus T$ is so large that any optimal s-plex cluster vertex deletion set would delete $H \cap T$ to disconnect the s-plex $H \setminus T$ from the rest of the graph, then we shrink $H \setminus T$ such that any optimal s-plex cluster vertex deletion set would still delete $H \cap T$.

an s-plex cluster vertex deletion set S' of size at most k for G. Since H induces an s-plex, it does not contain all vertices of F. Hence, F contains a vertex $v \in X$. Moreover, because H is X-separated, F contains an edge $\{v,w\}$ with $w \in H \cap T$. Since $G[H]$ is an s-plex, also $G[H \setminus \{u\}$ is an s-plex and from the preconditions of Reduction Rule 3 and from $|H \cap T| \geq 1$, we conclude

$$\begin{aligned} |H \setminus \{u\}| &= |H \cap T| + |H \setminus T| - 1 \\ &\geq |H \cap T| + |H \cap T| + 2s - 3 \\ &\geq 2s - 1. \end{aligned}$$

By Lemma 3, $G[H \setminus \{u\}]$ is connected. It follows that all vertices in $H \setminus \{u\}$, which contains w, are connected to v, since v and w are adjacent. Furthermore, the at least $|H \cap T| + 2s - 3$ vertices in $H \setminus (T \cup \{u\})$ are nonadjacent to v, because H is X-separated. Since $v,w \notin S$, the s-plex cluster vertex deletion set S for $G - \{u\}$ contains all but at most $2s-3$ vertices from $H \setminus (T \cup \{u\})$, thus $|S \cap (H \setminus T)| \geq |H \cap T|$. Consider the set $S' := (S \setminus (H \setminus T)) \cup (H \cap T)$. Since $G - S'$ is the s-plex cluster graph $G - (S \cup (H \cap T))$ with an additional connected component induced by $H \setminus T$ (containing u and inducing an s-plex), the set S' is an s-plex cluster vertex deletion set for G. Moreover, one has $|S'| \leq |S| \leq k$, implying that (G,k) is a yes-instance.

For the running time, observe that we can construct $\mathscr{H}(X)$ in $O(n+m)$ time. If for an $H \in \mathscr{H}(X)$ we find that all vertices in $H \setminus T$ are nonadjacent to X, then apply Reduction Rule 3 to H until $|H \setminus T| \leq |H \cap T| + 2s - 3$; deleting vertices from a graph takes $O(n+m)$ time in total. □

With Reduction Rule 3, the number of vertices in $H \setminus T$ is bounded from above by $|H \cap T| + 2s - 3$ for each X-separated set $H \in \mathscr{H}(X)$. It remains to shrink the size of non-X-separated sets.

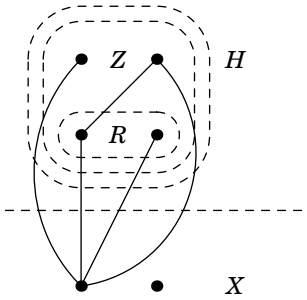


Fig. 6: A redundant subset R of a set $H \in \mathscr{H}(X)$ is shown. The redundant set R is contained in an X-module Z containing those vertices from H that are nonadjacent to at least one vertex in R. Note that all vertices in Z have the same neighbors in X and that vertices in R might be nonadjacent to each other.

4.3.2 Shrinking non-X-separated sets

Our goal is to find vertices in a non-X-separated set $H \in \mathscr{H}(X)$ that can safely be removed by a data reduction rule. To this end, we exploit that in a non-X-separated set $H \in \mathscr{H}(X)$, almost all vertices have the same neighbors in X by Lemma 4(3). This makes a large set of vertices in H "equivalent", so that many of them may be removed by data reduction rules. To formalize this, we capture the vertices having the same neighbors in X in an *X-module*:

Definition 3 For a vertex set $X \subseteq V$, we call a vertex set $Z \subseteq V$ an *X-module* if any two vertices $u, v \in Z$ satisfy $N(u) \cap X = N(v) \cap X$.

On Definition 3, we base the (quite technical) definition of redundant vertex sets, which contain "candidate" vertices for removal. The definition is illustrated in Figure 6.

Definition 4 Let $H \in \mathscr{H}(X)$. We call a subset $R \subseteq H$ *redundant* if there is an X-module Z with $R \subseteq Z \subseteq H$ that contains those vertices from H that are nonadjacent to at least one vertex in R.

Employing Definition 4, we state the following data reduction rule. After its correctness proof, the computation of redundant sets is described.

Reduction Rule 4 *Let $R \subseteq H$ be a redundant subset of some $H \in \mathscr{H}(X)$. If $|R| > k + 2s - 1$, then choose an arbitrary vertex from R and remove it from G.*

Lemma 6 *Reduction Rule 4 is correct.*

Proof Assume that for a vertex set $H \in \mathscr{H}(X)$ Reduction Rule 4 chooses to remove $u \in R$ from G, where R is a redundant subset of H. Let G' denote the graph $G - \{u\}$. We have to show that (G, k) is a yes-instance if and only if (G', k') is a yes-instance. Obviously, if (G, k) is a yes-instance, then (G', k) is also a yes-instance. If (G', k) is a yes-instance, then there is an s-plex cluster vertex deletion set S with $|S| \leq k$ for G'. We now show that S is also an s-plex cluster vertex deletion set for G, implying that (G, k) is a yes-instance. Assume for the purpose of contradiction that $G - S$ contains a forbidden induced subgraph. Because $G' - S$ is an s-plex cluster graph, $G - S$ contains a forbidden induced subgraph F containing u. In the subsequent arguments for the contradiction, we use the following two claims, which are proven afterwards.

Claim 1 In $G' - S$, the vertices of $F - \{u\}$ are connected to all vertices in $H \setminus (S \cup \{u\})$.

Claim 2 For an X-module Z with $R \subseteq Z \subseteq H$, if a vertex v of F is nonadjacent to a vertex $w \in Z \setminus S$, then $v \in H \setminus S$.

Now, we show the contradiction using Claim 1 and Claim 2. Because F is a forbidden induced subgraph in $G - S$, it contains a vertex v that is nonadjacent to a set W of s other vertices in F. If $u \notin \{v\} \cup W$, then Claim 1 shows that the vertices in $\{v\} \cup W$ are connected to all vertices in $H \setminus (S \cup \{u\})$. Thus, the vertices in $\{v\} \cup W$ would exist in $G' - S$ and would be connected. This contradicts $G' - S$ being an s-plex cluster graph. Thus, we have $u \in \{v\} \cup W$.

We prove the cases $u = v$ and $u \in W$ separately. For both cases, one needs to observe that $G[H \setminus S]$ is connected: because $G[H]$ is an s-plex, also $G[H \setminus S]$ is an s-plex. Moreover, by the precondition for the removal of u from G by Reduction Rule 4, we have $|H| > k + 2s - 1$. Since $|S| \leq k$, it holds that $|H \setminus S| > 2s - 1$. By Lemma 3, $G[H \setminus S]$ is connected.

Now, first assume that $u = v$. That is, the vertex $u \in R$ is nonadjacent to W. From Claim 2, one can conclude that $W \subseteq H \setminus S$. Because also $u \in H \setminus S$, this contradicts the graph $G[H \setminus S]$ being an s-plex. Thus, $u \in W$, and it follows that u is nonadjacent to v. From Claim 2, we conclude that $v \in H \setminus S$. By Definition 4, there is an X-module Z with $R \subseteq Z \subseteq H$ and $v \in Z$. Because the vertex $v \in Z$ is nonadjacent to W, the vertices in W are in $H \setminus S$ by Claim 2. Because also v is in $H \setminus S$, this again contradicts $G[H \setminus S]$ being an s-plex. We conclude that $G - S$ is an s-plex cluster graph.

It remains to show Claim 1 and Claim 2. For the proof of both claims, one first needs to observe that, because $R \setminus (S \cup \{u\}) \subseteq H$ and because $G[H]$ is an s-plex, also $G[R \setminus (S \cup \{u\})]$ is an s-plex. Second, since $|S| \leq k$ and because of the precondition for the removal of u from R by Reduction Rule 4, we have $|R \setminus (S \cup \{u\})| \geq 2s - 1$. Hence, by Lemma 3, $G[R \setminus (S \cup \{u\})]$ is connected. Analogously it follows that $G[H \setminus (S \cup \{u\})]$ is connected.

We first show Claim 1. Let v be a vertex of $F - \{u\}$. Because F is connected, there is a path in $G - S$, connecting v to u. This path uses a neighbor w of u (possibly, $v = w$). Now, distinguish between the two cases $w \in H \setminus S$ and $w \notin H \setminus S$. If $w \in H \setminus S$, then Claim 1 follows from the connectedness of $G[H \setminus (S \cup \{u\})]$. If $w \notin H \setminus S$, then $w \notin H$. Because w is a neighbor of $u \in R$, we have $w \in X$ and, because R is redundant, all vertices in $R \setminus (S \cup \{u\})$ are neighbors of w, showing Claim 1.

We now show Claim 2. For the purpose of contradiction, assume that there is an X-module Z with $R \subseteq Z \subseteq H$ and that a vertex $v \notin H \setminus S$ of F is nonadjacent to $w \in Z \setminus S$. Clearly, because v is in $G - S$, this implies $v \notin H$ and, hence, $v \neq u$. We first show that v is nonadjacent to Z. If v is adjacent to Z, then $v \in X$, because $v \notin H$. Because $w \in Z$ and Z is an X-module, w is adjacent to v, contradicting our assumption. Therefore, v is nonadjacent to Z and, in particular, nonadjacent to $R \setminus S$. According to Claim 1, the at least $2s - 1$ vertices in $R \setminus (S \cup \{u\})$ are connected but nonadjacent to v in $G' - S$. This implies that there is a forbidden induced subgraph in $G' - S$, contradicting $G' - S$ being an s-plex cluster graph. □

With the completion of the correctness proof of Reduction Rule 4, we obtained a data reduction rule for non-X-separated sets in $\mathscr{H}(X)$. Since applying Reduction Rule 4 requires the existence of a large redundant subset of some $H \in \mathscr{H}(X)$, we shall in the following present a way to compute such redundant subsets. This finally leads to the proof of the problem kernel for s-Plex Cluster Vertex Deletion.

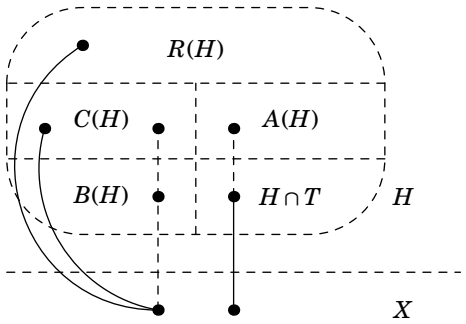


Fig. 7: Illustration of the sets $A(H)$, $B(H)$, $C(H)$, and $R(H)$. Solid lines indicate edges and dashed lines "non-edges". Note that the sets $A(H)$, $B(H)$, $C(H)$, and $H \cap T$ might have pairwise non-empty intersections (which is not relevant for our arguments); to keep the figure simple, they are drawn without intersections.

4.3.3 Finding Redundant Sets

We now show how to efficiently find a redundant subset $R \subseteq H$ for a non-X-separated set $H \in \mathscr{H}(X)$. To this end, recall that X is an s-plex cluster vertex deletion set and that T is a tidying set. Intuitively, we first build vertex subsets of H that potentially violate Definition 4 for redundant sets. Then, we show that the remaining vertices of H constitute a redundant set. See Figure 7 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 at least one vertex in $B(H)$. Let $\bar{R}(H) := A(H) \cup B(H) \cup C(H)$ and let the remaining vertices of H be $R(H) := H \setminus (\bar{R}(H) \cup T)$.

Lemma 7 *For each $H \in \mathscr{H}(X)$, 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.*

Proof To prove that $R(H)$ is redundant, one has to show that there is a set $Z(H)$, $R(H) \subseteq Z(H) \subseteq H$, that contains those vertices of H that are nonadjacent to at least one vertex in $R(H)$ and such that $Z(H)$ is an X-module, that is, for $u, v \in Z(H)$, $N(u) \cap X = N(v) \cap X$.

Consider the set $Z(H) := \{u \in H \setminus T \mid N(u) \cap X = N(H \setminus T) \cap X\}$. For any two vertices $u, v \in Z(H)$, we have $N(u) \cap X = N(H \setminus T) \cap X = N(v) \cap X$. Thus, the set $Z(H) \subseteq H$ is an X-module. It remains to show that $R(H) \subseteq Z(H)$ and that $Z(H)$ contains those vertices of H that are nonadjacent to at least one vertex in $R(H)$. To show that a vertex u is in $Z(H)$, one has to show $u \in H \setminus T$ and $N(H \setminus T) \cap X \subseteq N(u) \cap X$ (and $N(u) \cap X \subseteq N(H \setminus T) \cap X$, which follows directly from $u \in H \setminus T$).

We first show that $R(H) \subseteq Z(H)$. Obviously, $R(H) \subseteq H \setminus T$. For a vertex $w \in N(H \setminus T) \cap X$, each vertex $u \in R(H)$ is adjacent to w. Otherwise, u would be in $B(H)$. This implies $N(H \setminus T) \cap X \subseteq N(u) \cap X$ and $u \in Z(H)$.

Now, we show that $Z(H)$ contains those vertices of H that are nonadjacent to at least one vertex in $R(H)$. Assume that there is a vertex $u \in R(H)$ and a vertex $w \in H$ nonadjacent to u. Because $u \notin A(H)$, we have $w \notin T$. For a vertex $v \in N(H \setminus T) \cap X$,

the vertex w is adjacent to v; otherwise, $w \in B(H)$ and, therefore, $u \in C(H)$. Thus, $N(H \setminus T) \cap X \subseteq N(w) \cap X$ and $w \in Z(H)$.

To prove the size bound, note that because H induces an s-plex, it holds that $|A(H)| \in O(s \cdot |H \cap T|)$ and $|C(H)| \in O(s \cdot |B(H)|)$. According to Lemma 4(3), if a vertex $v \in X$ is adjacent to $H \setminus T$, then there are at most $2s-3$ vertices in $H \setminus T$ nonadjacent to v. Thus, $|B(H)| \in O(s \cdot |N(H \setminus T) \cap X|)$. □

Lemma 8 *For all $H \in \mathscr{H}(X)$, the sets $R(H)$ can be computed in $O(n^2)$ time.*

Proof One can construct $\mathscr{H}(X)$ in $O(n+m)$ time. Then, scan each set $H \in \mathscr{H}(X)$ in three phases. First, collect vertices belonging to $H \cap T$ and for each vertex in $H \setminus T$, remember its neighbors in X to belong to $N(H \setminus T) \cap X$. Second, put vertices that are nonadjacent to some vertex in $H \cap T$ or $N(H \setminus T) \cap X$ into the sets $A(H)$ and $B(H)$, respectively. Third, analogously, construct $C(H)$ from $B(H)$. Finally, we output all vertices of H that are neither in $A(H), B(H), C(H)$ nor in T. It is easy to see that all steps work in $O(n^2)$ time. □

By Lemma 8, we can compute in $O(n^2)$ time the sets $R(H)$ and shrink them using Reduction Rule 4 so that $|R(H)| \leq k + 2s - 1$. The number of vertices in $H \setminus (R(H) \cup T)$ is $O(s \cdot |H \cap T| + s^2 \cdot |N(H \setminus T) \cap X|)$ due to Lemma 7. This shows:

Lemma 9 *Reduction Rule 4 can be applied in $O(n^2)$ time so that, afterwards, for each non-X-separated set $H \in \mathscr{H}(X)$, one has $|H \setminus T| \in O(s \cdot |H \cap T| + s^2 \cdot |N(H \setminus T) \cap X| + k)$.*

4.4 Problem Kernel Size

Now we have all ingredients to prove the size of our problem kernel for s-Plex Cluster Vertex Deletion. In the next section, we will improve the running time of our kernelization algorithm to $O(ksn^2)$.

Theorem 1 *s-Plex Cluster Vertex Deletion admits a problem kernel of $O(k^2 s^3)$ vertices, which can be computed in $O(ksn^{s+\sqrt{s}+1})$ time.*

Proof It is straightforward to verify that Approximation Step, Tidying Step, and Shrinking Step work in $O(ksn^{s+\sqrt{s}+1})$ time in total. It remains to prove the kernel size. Recall that $|X| \in O(ks)$ and $|T| \in O(k^2 s^2)$, where X is the approximate s-plex cluster vertex deletion set and where T is the tidying set. To bound the number of vertices that remain in $G - (T \cup X)$ after applying all data reduction rules, we have to bound the number of vertices in $H \setminus T$ for all $H \in \mathscr{H}(X)$. Since Reduction Rule 2 has been applied, each $H \in \mathscr{H}(X)$ is adjacent to X and, therefore, either X-separated or non-X-separated. For an X-separated set H one has $|H \setminus T| \leq |H \cap T| + 2s - 3$ (Reduction Rule 3). Moreover, since each X-separated set contains at least one vertex of T, there are at most $|T|$ many X-separated sets in $\mathscr{H}(X)$. Hence, the number of vertices in $H \setminus T$ for all X-separated sets is at most $|T| + |T|(2s-3) \in O(k^2 s^3)$.

It remains to count the vertices contained in non-X-separated sets. To this end, we partition the approximate s-plex cluster vertex deletion set X into

$$X_1 := \{v \in X \mid \text{ there is exactly one set } H \in \mathscr{H}(X) \text{ such that } H \setminus T \text{ is adjacent to } v\}$$

and $X_2 := X \setminus X_1$. According to Lemma 4(2), each non-X-separated set $H \in \mathscr{H}(X)$ for which $H \setminus T$ is adjacent to a vertex $v \in X_2$ contains $O(s)$ vertices. From $|X_2| \in O(ks)$, we

conclude that there are at most $O(ks^2)$ vertices in non-X-separated sets $H \in \mathscr{H}(X)$ for which $H \setminus T$ is adjacent to a vertex $v \in X_2$.

It finally remains to count the vertices in sets $H \in \mathscr{H}(X)$ for which $H \setminus T$ is adjacent to only vertices in X_1. Let these sets be contained in the set $\tilde{\mathscr{H}} \subseteq \mathscr{H}(X)$. We first show that

$$\sum_{H \in \tilde{\mathscr{H}}} |N(H \setminus T) \cap X| = |X_1| \qquad \text{and} \qquad |\tilde{\mathscr{H}}| \leq |X_1|. \tag{1}$$

Let $H \in \tilde{\mathscr{H}}$ be a set such that $H \setminus T$ is only adjacent to vertices in X_1. For a vertex $v \in X_1$ being adjacent to $H \setminus T$, there is by definition of X_1 no other set $H' \in \mathscr{H}(X)$ such that $H' \setminus T$ is adjacent to v. Thus, if we count the number of vertices in $N(H \setminus T) \cap X$ for all $H \in \tilde{\mathscr{H}}$, then we count every vertex $v \in X_1$ exactly once. This proves the first relation. Now, for each non-X-separated $H \in \tilde{\mathscr{H}}$, there is at least one vertex $v \in X_1$ such that $H \setminus T$ is adjacent to v. Thus, $|\tilde{\mathscr{H}}| \leq |X_1|$. We now count the vertices in the sets in $\tilde{\mathscr{H}}$, which, according to Lemma 9, Reduction Rule 4 has shrunk to

$$O\Big(\sum_{H \in \tilde{\mathscr{H}}} (s \cdot |H \cap T| + s^2 \cdot |N(H \setminus T) \cap X| + k)\Big).$$

The sets in $\tilde{\mathscr{H}}$ are pairwise disjoint, implying $\sum_{H \in \tilde{\mathscr{H}}} s \cdot |H \cap T| \leq s \cdot |T|$. Thus, this term is

$$\begin{aligned} &O\Big(\sum_{H \in \tilde{\mathscr{H}}} (s^2 \cdot |N(H \setminus T) \cap X| + k) + s \cdot |T|\Big) \\ &= O(s^2 \cdot |X_1| + k \cdot |X_1| + s \cdot |T|), \end{aligned}$$

where the latter equality follows from Equation 1. Exploiting $|X_1| \in O(ks)$ and $|T| \in O(k^2 s^2)$, it follows that $O(k^2 s^3)$ vertices are in sets in $\tilde{\mathscr{H}}$. □

5 Speeding up the Kernelization Algorithm

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 1 is the computation of the tidying set T in Section 4.2, which relies on finding for each $v \in X$ a maximal set $\mathscr{F}(v)$ of forbidden induced subgraphs that pairwise intersect exactly in v, thus taking $O(ksn^{s+\sqrt{s}+1})$ time. The maximality of $\mathscr{F}(v)$ was used to prove that $G - T$ satisfies the tidiness property. This, in turn, helped us to show that T satisfies the properties stated in Lemma 4. However, we obtain a much faster kernelization if we do not demand that $\mathscr{F}(v)$ is maximal; rather, this section shows how to compute, in $O(ksn^2)$ time, a bounded-size set T that also satisfies the properties in Lemma 4.

Lemma 10 *Let X be an s-plex cluster vertex deletion set. For all $v \in X$, vertex sets $T(v)$ satisfying the following properties can be found in $O(ksn^2)$ time:*

1. *There are at most s sets $H \in \mathscr{H}(X)$ such that $H \setminus T(v)$ is adjacent to v.*
2. *If there is more than one set $H \in \mathscr{H}(X)$ such that $H \setminus T(v)$ is adjacent to v, then each such set H satisfies $|H \setminus T(v)| \leq 2s - 2$.*
3. *If there is a set $H \in \mathscr{H}(X)$ such that $H \setminus T(v)$ is adjacent to v, then v is nonadjacent to at most $2s - 3$ vertices in $H \setminus T(v)$.*

Proof Given the s-plex cluster vertex deletion set X, first one constructs $\mathscr{H}(X)$, which is needed to efficiently construct the sets $T(v)$. Constructing $\mathscr{H}(X)$ can be done in $O(n+m)$ time using breadth-first search, additionally storing for each vertex to which set in $\mathscr{H}(X)$ (corresponding to a connected component of $G-X$) it belongs. Then, for each $v \in X$, execute the following three steps:

(1) For each $u \in N(v) \setminus T(v)$, if there is a set $W \subseteq N(v) \setminus T(v)$ of s neighbors of v such that u is nonadjacent to W, then add u and the vertices in W to $T(v)$. Clearly, $G[\{v,u\} \cup W]$ forms a forbidden induced subgraph. Repeat this step until there is no such set W of s neighbors.

(2) For each $u \in N(v) \setminus T(v)$, get the set $H \in \mathscr{H}(X)$ with $u \in H$. If there is a set $W \subseteq \mathscr{H}(X) \setminus T(v)$ with $|W| = 2s-1$ and a vertex $w \in N(v) \setminus (T(v) \cup H)$, then add the vertices in W and w to $T(v)$. Continue with the next $u \in N(v) \setminus T(v)$. By Lemma 3, an s-plex with $2s-1$ vertices is a connected graph. Hence, $G[W \cup \{u\}]$, being an induced subgraph of the s-plex $G[H]$, is connected and $G[\{v,u,w\} \cup W]$ forms a forbidden induced subgraph because w is connected but nonadjacent to all vertices in W.

(3) For each neighbor $u \in N(v) \setminus T(v)$, consider the set $H \in \mathscr{H}(X)$ with $u \in H$. If there is a set $W \subseteq H \setminus T(v)$ with $|W| = 2s-2$ such that W is nonadjacent to v, then add u and the vertices in W to $T(v)$. Clearly, $G[\{v,u\} \cup W]$ forms a forbidden induced subgraph since v is nonadjacent but connected to all vertices in W: this follows from Lemma 3 and $G[W \cup \{u\}]$ being an s-plex. Repeat this step until there is no such set W of size $2s-2$.

Obviously, after that, since neither step (1), nor step (2), nor (3) apply, $T(v)$ satisfies the properties stated by the lemma. Clearly, for each fixed $v \in X$, all steps work in $O(n^2)$ time. The stated time bound of $O(ksn^2)$ then follows by exploiting $|X| \in O(ks)$. □

The set $\bigcup_{v \in X} T(v)$ resulting from Lemma 10 satisfies exactly the properties stated for the tidying set T in Lemma 4. Hence, the expensive computation of the tidying set T in Section 4.2 can be replaced by the computation described in the proof of Lemma 10. In order to bound the size of the set $\bigcup_{v \in X} T(v)$, a slightly modified version of Reduction Rule 1 allows us to remove vertices v from X for which $T(v)$ is large:

Reduction Rule 1' *If there is a vertex $v \in X$ such that $|T(v)| > 2ks$, then delete v from G and from X and decrement the number k of allowed vertex deletions by one.*

Lemma 11 *Reduction Rule 1' is correct and can be exhaustively applied in $O(n+m)$ time.*

Proof For each $v \in X$, the three steps in the proof of Lemma 10 "collect" vertices of forbidden induced subgraphs that pairwise intersect exactly in v. For each found forbidden induced subgraph, at most $2ks$ vertices are added to $T(v)$. Hence, if $|T(v)| > 2ks$, then more than k forbidden induced subgraphs pairwise intersect exactly in v. Hence, v must be contained in any s-plex cluster vertex deletion set of size at most k. The running time follows analogously to that of Reduction Rule 1. □

Finally, after applying Reduction Rule 1', for each vertex $v \in X$, we have $|T(v)| \leq 2ks$. Hence, replacing Reduction Rule 1 by Reduction Rule 1' and setting $T := \bigcup_{v \in X} T(v)$ instead of using the expensive computation of T in the Tidying Step in Section 4.2, we obtain the central theorem of our work:

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

6 Conclusion

There are two main performance aspects of problem kernelization—one is the size of the problem kernel and the other one is its (polynomial) running time. *Both* need to be optimized. Studying s-PLEX CLUSTER VERTEX DELETION and introducing the approximation and tidying kernelization method, we contributed to both. Our results are based on linking kernelization with polynomial-time approximation, obtaining preprocessing rules to detect combinatorial input structures to be exploited by effective data reduction rules. The approximation and tidying kernelization method likely applies to other vertex deletion problems whose goal graphs can be 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 [19].

As to future work, it remains to provide further applications for kernelization through approximation and tidying. It also seems interesting to test the practical usefulness of the developed data reduction rules. Finally, it would be desirable to start a general study under which conditions fixed-parameter tractable vertex deletion problems possess polynomial-size kernels. Known candidates having no polynomial-size kernel (unless NP $\subseteq$ coNP/poly, implying a collapse of the polynomial-time hierarchy to the third level [4, 5]) are CONNECTED VERTEX COVER and CAPACITATED VERTEX COVER [9].

In the field of graph-based data clustering, it is desirable to compare the clustering results using the s-plex concept with clustering results using other cluster concepts, like s-clubs [2, 24] (subgraphs with diameter at most s) or overlapping clusters [11]. For s-PLEX CLUSTER VERTEX DELETION, it remains to find a fixed-parameter algorithm that is faster than finding forbidden induced subgraphs and branching into all possibilities of deleting one of a forbidden induced subgraph's vertices.

Acknowledgments We are grateful to two anonymous referees of *Algorithmica* for comments helping to improve the presentation.

References

1. F. N. Abu-Khzam. A kernelization algorithm for d-Hitting Set. *Journal of Computer and System Sciences*, 76(7):524 – 531, 2010.
2. R. D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3:3–113, 1973.
3. B. Balasundaram, S. Butenko, and I. V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 2011. To appear.
4. H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation (IWPEC '09)*, volume 5917 of *Lecture Notes in Computer Science*, pages 17–37. Springer, 2009.
5. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8): 423–434, 2009.
6. 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. *Nature Genetics*, 37(3):233–242, 2005.

7. 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. *Journal of Infectious Diseases*, 196:1517–1527, 2007.
8. J. Díaz and D. M. Thilikos. Fast FPT-algorithms for cleaning grids. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS '06)*, volume 3884 of *Lecture Notes in Computer Science*, pages 361–371. Springer, 2006.
9. M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *Proceedings of the 36th International Colloquium on Automata, Languages, and Programming (ICALP '09)*, volume 5555 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2009.
10. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
11. M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Graph-based data clustering with overlaps. *Discrete Optimization*, Available electronically, 2010.
12. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
13. J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.
14. J. Guo, H. Moser, and R. Niedermeier. Iterative compression for exactly solving NP-hard minimization problems. In *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 65–80. Springer, 2009.
15. J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. A more relaxed model for graph-based data clustering: s-plex cluster editing. *SIAM Journal on Discrete Mathematics*, 24(4):1662–1683, 2010.
16. F. Hüffner, R. Niedermeier, and S. Wernicke. Techniques for practical fixed-parameter algorithms. *The Computer Journal*, 51(1):7–25, 2008.
17. F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory of Computing Systems*, 47(1):196–217, 2010.
18. P. G. Kolaitis and M. N. Thakur. Logical definability of NP optimization problems. *Information and Computation*, 115(2):321–353, 1994.
19. S. Kratsch. Polynomial kernelizations for MIN $\mathrm{F}^{+}\Pi_1$ and MAX NP. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS '09)*, pages 601–612. IBFI Dagstuhl, Germany, 2009.
20. J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.
21. D. Marx and I. Schlotter. Parameterized graph cleaning problems. *Discrete Applied Mathematics*, 157(15):3258–3267, 2009.
22. B. McClosky and I. Hicks. Combinatorial algorithms for the maximum k-plex problem. *Journal of Combinatorial Optimization*, 2010. Available electronically.
23. 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 *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES '07)*, pages 861–870. IEEE Computer Society Press, 2007.

24. R. J. Mokken. Cliques, clubs and clans. *Quality & Quantity*, 13:161–173, 1979.
25. H. Moser, R. Niedermeier, and M. Sorge. Algorithms and experiments for clique relaxations—finding maximum s-plexes. In *Proceedings of the 8th International Symposium on Experimental Algorithms (SEA '09)*, volume 5526 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2009.
26. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
27. B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.
28. S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.
29. S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978.
30. R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1–2):173–182, 2004.
31. B. Wu and X. Pei. A parallel algorithm for enumerating all the maximal k-plexes. In *Emerging Technologies in Knowledge Discovery and Data Mining*, volume 4819 of *Lecture Notes in Artificial Intelligence*, pages 476–483. Springer, 2007.