A MORE RELAXED MODEL FOR GRAPH-BASED DATA CLUSTERING: S-PLEX CLUSTER EDITING*

JIONG GUO[†§], CHRISTIAN KOMUSIEWICZ[‡¶], ROLF NIEDERMEIER[‡], AND JOHANNES UHLMANN[‡‖]

Abstract. We introduce the s-Plex Cluster Editing problem as a generalization of the well-studied Cluster Editing problem, both being NP-hard and both being motivated by graph-based data clustering. Instead of transforming a given graph by a minimum number of edge modifications into a disjoint union of cliques (this is Cluster Editing), the task in the case of s-Plex Cluster Editing is to transform a graph into a cluster graph consisting of a disjoint union of so-called s-plexes. Herein, an s-plex is a vertex set S inducing a subgraph in which every vertex has degree at least $|S| - s$. Cliques are 1-plexes. The advantage of s-plexes for $s \geq 2$ is that they allow to model a more relaxed cluster notion (s-plexes instead of cliques), better reflecting inaccuracies of the input data. We develop a provably effective preprocessing based on data reduction (yielding a so-called problem kernel), a forbidden subgraph characterization of s-plex cluster graphs, and a depth-bounded search tree which is used to find optimal edge modifications sets. Altogether, this yields efficient algorithms in case of moderate numbers of edge modifications, this often being a reasonable assumption under a maximum parsimony model for data clustering.

Key words. NP-hard problems, exact algorithms, fixed-parameter tractability, data reduction, graph modification, k-plex, dense subgraphs, forbidden subgraph characterization.

AMS subject classifications. 05C85, 68R10, 68W99.

1. Introduction.

The purpose of a clustering algorithm is to group together a set of (many) objects into a relatively small number of clusters such that the elements inside a cluster are highly similar to each other whereas elements from different clusters have low or no similarity. There are numerous approaches to clustering and "there is no clustering algorithm that can be universally used to solve all problems" [36]. One prominent line of attack is to use methods based on graph theory [30, 32]. In this line, extending and complementing previous work on cluster graph modification problems, we introduce the new edge modification problem s-Plex Cluster Editing.

In the context of graph-based clustering, data items are represented as vertices and there is an undirected edge between two vertices if and only if the interrelation between the two corresponding items exceeds some threshold value. Clustering with respect to such a graph then means to partition the vertices into sets where each set induces a dense subgraph (that is, a *cluster*) of the input graph whereas there are only few edges between the vertices of different clusters. In this scenario, the algorithmic task then typically is to transform the given graph into a so-called cluster graph by a minimum number of graph modification operations [32]. Herein, a *cluster graph* is

*A preliminary version appeared in the Proc. of the 5th International Conference on Algorithmic Aspects in Information and Management (AAIM '09), volume 5564 in LNCS, pages 226–239, Springer 2009. Note that in the conference version the problem was just called s-Plex Editing, whereas we later found s-Plex Cluster Editing to be the more appropriate naming.

[†]Universität des Saarlandes, Campus E 1.4, D-66123 Saarbrücken, Germany. E-mail: jguo@mmci.uni-saarland.de

[‡]Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany. E-mail: {c.komus, rolf.niedermeier, johannes.uhlmann}@uni-jena.de

[§]Supported by Excellence Cluster on Multimodal Computing and Interaction (MMCI). Main work was done while the author was with the Friedrich-Schiller-Universität Jena.

[¶]Supported by a PhD fellowship of the Carl-Zeiss-Stiftung and the DFG, research project PABI, NI 369/7.

[‖]Supported by the DFG, research project PABI, NI 369/7.

a graph where all connected components form clusters and a graph modification is to insert or delete an edge. One of the most prominent problems in this context is the NP-hard Cluster Editing problem (also known as Correlation Clustering) [32, 2], where, given a graph G and an integer $k \geq 0$, one wants to transform G into a graph whose connected components all are cliques, using at most k edge insertions and deletions. In this work, with the NP-hard s-Plex Cluster Editing problem, we study a more relaxed and often presumably more realistic variant of Cluster Editing: Whereas in the case of Cluster Editing the clusters shall be cliques, in the case of s-Plex Cluster Editing we only demand them to be s-plexes. A vertex subset $S \subseteq V$ of a graph $G = (V, E)$ is called *s-plex* if the minimum vertex degree in the induced subgraph $G[S]$ is at least $|S| - s$. Note that a clique is nothing but a 1-plex. Replacing cliques by s-plexes for some integer $s \geq 2$ allows one to reflect the fact that most real-world data are somewhat "spurious" and so the demand for cliques may be overly restrictive in defining what a cluster shall be (also see [8, 31] concerning criticism of the overly restrictive nature of the clique concept).

Problem formulation. In the following, we call a graph an *s-plex cluster graph* if all its connected components are s-plexes.

s-Plex Cluster Editing
Input: An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Can G be modified by up to k edge deletions and insertions into an s-plex cluster graph?

Indeed, seen as an optimization problem, the goal is to minimize the number of edge modifications.[1] Note that 1-Plex Cluster Editing is the same as Cluster Editing. Compared to Cluster Editing, s-Plex Cluster Editing with $s \geq 2$ is a more flexible tool for graph-based data clustering: For increasing s, the number of edge modifications should decrease.[2] This important advantage of s-Plex Cluster Editing reflects the observation that fewer edge modifications mean that we introduce fewer "errors" into our final cluster solution, because the computed s-plex cluster graph is closer to the original data. This is in accordance with the natural hypothesis that the less one perturbs the input graph the more robust and plausible the achieved clustering is (maximum parsimony principle, also see Böcker et al. [5] for making this point in terms of Cluster Editing). Figure 1.1 presents a simple example comparing Cluster Editing (that is, 1-Plex Cluster Editing) with 2-Plex Cluster Editing and 3-Plex Cluster Editing in terms of the (number of) necessary edge modifications.

Previous work and motivation. Our work combines two lines of research, one dealing with the s-plex concept and the other one dealing with the NP-hard Cluster Editing problem. The s-plex concept was introduced in 1978 by Seidman and Foster [31] in the context of social network analysis. Recently, a number of theoretical and experimental studies explored (and confirmed) the usefulness of s-plexes in various contexts [1, 10, 22, 24, 25]. These studies exploit that s-plexes are a more relaxed notion of the concept of dense (sub)graph than cliques are. Concerning computational complexity, in analogy to finding maximum-cardinality cliques also finding maximum-cardinality s-plexes is NP-hard [1] and further hardness results in analogy to clique finding hold as well [22]. Wu and Pei [35] considered the problem of enumer-

[1] All algorithms in this work can easily solve the optimization version of s-Plex Cluster Editing, where k is minimized.

[2] The number of edge modifications is non-increasing for increasing s, since an s-plex is also an $(s + 1)$-plex.

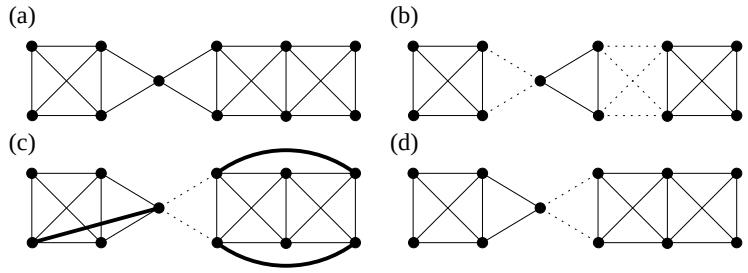

FIG. 1.1. *An example for different optimal (numbers of) modifications that are applied to (a) an input graph using (b)* CLUSTER EDITING *(equivalently,* 1-PLEX CLUSTER EDITING*), (c)* 2-PLEX CLUSTER EDITING*, and (d)* 3-PLEX CLUSTER EDITING*. Deleted edges are dashed, inserted edges are bold.*

ating all maximal s-plexes. See also Cohen et al. [9] for related work on enumerating all maximal induced subgraphs for (connected-)hereditary graph properties.

CLUSTER EDITING, which uses cliques for defining clusters, is one of the most intensively studied problems in graph-based data clustering. Due to its NP-hardness, it has been considered from the viewpoints of polynomial-time approximability as well as parameterized algorithmics. As to approximability, the currently best known approximation factor is 2.5 [37]. Considering the parameter k defined as the number of allowed edge modifications, a search tree of size $O(1.83^k)$ [5] has been developed and several studies concerning provably effective preprocessing by polynomial-time data reduction (which is called problem kernelization in the context of parameterized algorithmics [12, 15, 26]) have been performed [7, 13, 16, 17, 28]. The advantage of fixed-parameter algorithms is that, other than approximation algorithms, they find minimum-cardinality sets of editing operations. Fixed-parameter algorithms have led to several successful experimental studies mainly in the context of biological network analysis [5, 6, 11, 29, 34]. Thus, fixed-parameter algorithms combined with clever heuristics seem to be a locomotive for practically solving instances of CLUSTER EDITING. However, CLUSTER EDITING is NP-hard and, as a consequence, the parameterized algorithms only run fast in case of moderate values of the parameter k, the number of allowed edge editing operations. Hence, it is desirable to have the parameter k small not only for the sake of not too much perturbing the input graph but also for the sake of obtaining efficient solving algorithms.[3]

In summary, s-PLEX CLUSTER EDITING, so far unexplored, might become a valuable alternative to CLUSTER EDITING. To this end, this work provides a first theoretical study of s-PLEX CLUSTER EDITING. Despite its NP-hardness, we can deliver several algorithmic results, supporting the hope for a practically useful clique relaxation in the context of cluster graph modification problems.

Our contributions. In Section 3, we develop a polynomial-time preprocessing algorithm that allows to provably simplify input instances of s-PLEX CLUSTER EDITING to smaller ones. More specifically, the corresponding data reduction rules, given an instance $(G = (V, E), k)$ of s-PLEX CLUSTER EDITING with $s \geq 2$, in polynomial time construct an equivalent reduced instance[4] $(G' = (V', E'), k')$, $k' \leq k$, and, most

[3]The essential fact is that fixed-parameter algorithms have time complexity exponential in the parameter but polynomial in the overall input size, see Section 2 for more on that.

[4]By "equivalent" we mean that yes-instances are mapped to yes-instances and no-instances are mapped to no-instances, and, moreover, a solution set for the original instance can be easily con-

importantly, $|V'| \leq (8s^2 - 6) \cdot k + 8(s-1)^2$. In other words, the number of vertices of the reduced graph only depends on s and k (in fact, in case of s being a constant, it is linear in k), implying that if k is small then the data reduction will greatly simplify the instance basically without losing information. In terms of parameterized algorithmics, the reduced instance gives a so-called problem kernel (see Section 2 for more on that). Moreover, in Section 4, we provide a graph-theoretic characterization of s-plex cluster graphs by means of forbidden subgraphs. In particular, we obtain a linear-time recognition algorithm for s-plex cluster graphs for every constant s. This is of independent graph-theoretic interest and is also of decisive algorithmic use: Based on the forbidden subgraph characterization of s-plex cluster graphs, in Section 5, we show that s-Plex Cluster Editing can be solved in $O((2s + \lfloor\sqrt{s}\rfloor)^k \cdot s \cdot (|V| + |E|))$ time (which is linear for constant values of s and k) and, alternatively, in $O((2s + \lfloor\sqrt{s}\rfloor)^k + |V|^4)$ time.

2. Preliminaries. We only consider *undirected* graphs $G = (V, E)$, where V is the set of vertices and E is the set of edges. Throughout this paper, we set $n := |V|$ and $m := |E|$. For a graph G, we also use $V(G)$ and $E(G)$ to denote its vertex and edge sets, respectively. The *(open) neighborhood* $N_G(v)$ of a vertex $v \in V$ is the set of vertices that are adjacent to v in G. The *degree* of a vertex v, denoted by $\deg_G(v)$, is the cardinality of $N_G(v)$. For a set U of vertices, $N_G(U) := \bigcup_{v \in U} N_G(v) \setminus U$. We use $N_G[v]$ to denote the *closed* neighborhood of v, that is, $N_G[v] := N_G(v) \cup \{v\}$. For a set of vertices $V' \subseteq V$, the *induced subgraph* $G[V']$ is the graph over the vertex set V' with edge set $\{\{v, w\} \in E \mid v, w \in V'\}$. For $V' \subseteq V$ we use $G - V'$ as an abbreviation for $G[V \setminus V']$ and for a vertex $v \in V$ let $G - v$ denote $G - \{v\}$. A vertex $v \in V(G)$ is called a *cut-vertex* if $G - v$ has more connected components than G.

The main purpose of our work is to study the algorithmic tractability of s-Plex Cluster Editing. Unfortunately (but not surprisingly), it turns out to be NP-complete, excluding any hope for polynomial-time algorithms. The NP-completeness of s-Plex Cluster Editing with $s = 1$ (that is, Cluster Editing) was shown by Křivánek and Morávek [23]. A slightly modified version of a different NP-hardness proof by Shamir et al. [32] works for all $s \geq 2$. We omit the basically straightforward details.

Parameterized algorithmics [12, 15, 26] aims at a multivariate complexity analysis of problems without giving up the demand for finding optimal solutions. This is undertaken by studying relevant problem parameters and their influence on the computational hardness of problems. The hope lies in accepting the seemingly inevitable combinatorial explosion for NP-hard problems, but confining it to the parameter. Hence, the decisive question is whether a given parameterized problem is *fixed-parameter tractable (FPT)* with respect to a parameter k. In other words, one asks for the existence of a solving algorithm with running time $f(k) \cdot \text{poly}(n)$ for some computable function f. A core tool in the development of parameterized algorithms is polynomial-time preprocessing by *data reduction rules*, often yielding a *problem kernel* [4, 18]. Herein, the goal is, given any problem instance G with parameter k, to transform it in polynomial time into a new instance G' with parameter k' such that the size of G' is bounded from above by some function only depending on k, $k' \leq k$, and (G, k) is a yes-instance if and only if (G', k') is a yes-instance. We call a data reduction rule *correct* if the new instance after an application of this rule is a yes-instance if and only if the original instance is a yes-instance. We also employ search

structed from the solution set for the reduced instance.

trees for our parameterized algorithms. Search tree algorithms work in a recursive manner. The number of nodes in the corresponding tree is the number of recursion calls.

3. Data Reduction and Kernelization. In this section, we show that s-Plex Cluster Editing for $s \geq 2$ admits a problem kernel with at most $(8s^2-6)\cdot k+8(s-1)^2$ vertices. In particular, for the most relevant case of constantly bounded s-values, the problem kernel has a linear number of vertices. Since s-Plex Cluster Editing is a generalization of Cluster Editing, it is a natural idea to follow a kernelization strategy that leads to $O(k)$-vertex kernels for Cluster Editing [17, 7]. This strategy makes use of the central observation that for Cluster Editing the vertices of each resulting clique that are not incident to any modified edge form a so-called "critical clique" in the input graph, that is, they all have the same closed neighborhood. In order to obtain a kernel with $O(k)$ vertices, the crucial point is to bound the size of critical cliques in the input graph. To this end, one needs two observations: First, optimal edge modification sets never "split" a critical clique. This means that for every critical clique there is a clique in the resulting cluster graph entirely containing it. Second, if a critical clique is large enough compared to its neighborhood, then there must be a clique in the resulting cluster graph containing this critical clique and its neighbors but nothing else. Then, as long as the critical clique is large enough, the reduction rule can safely remove vertices of this critical clique from the input graph.

In contrast, an s-plex can have an unbounded number of vertices that have different neighborhoods. Therefore, we need a new concept other than critical cliques to represent a set of vertices that form an s-plex and that have weak connection to the remaining part of the graph. Moreover, if there is such an "almost-s-plex" P of large size, then we need to reduce the size of P. However, we cannot simply remove some vertices from P as in the Cluster Editing case, since the vertices in P have different neighborhoods in $V \setminus P$. Consequently, we need a more sophisticated data reduction rule (Reduction Rule 2) and the accompanying mathematical analysis requires new tools.

The first data reduction rule is obvious and its running time is $O(n+m)$:

Reduction Rule 1: Remove from G connected components that are s-plexes.

Our problem kernelization consists of only one further, technically complicated data reduction rule. Roughly speaking, the idea behind this rule is that a data reduction can be performed if there is a vertex with a "dense local environment" that is only loosely connected to the rest of the graph.

We start with explaining in more detail the purpose of introducing the second data reduction rule. Let G_{plex} denote the s-plex cluster graph resulting from applying a solution S with $|S| \leq k$ to the graph $G = (V, E)$ which is reduced with respect to the first rule, and let $K_1, \ldots, K_l$ be the s-plexes in G_{plex}. The vertex set V can be partitioned into two subsets, namely, X, the set of vertices that are endpoints of the edges modified by S, and $Y := V \setminus X$. For each s-plex K_i, let $X_i := X \cap K_i$ and $Y_i := Y \cap K_i$. Clearly, $|X| \leq 2k$. It remains to analyze $|Y|$. If $|Y_i| \leq (s-1)\cdot|X_i|$ or $|Y_i| \leq 2(s-1)$ for all i, we can conclude that there is a problem kernel with $O(k)$ vertices, since there can be at most $2k$ s-plexes in G_{plex}. However, for an arbitrary yes–instance, each Y_i can contain an unbounded number of vertices. Hence, a second data reduction rule is required. This rules is based on the following structural observations. Consider an s-plex K_i with $|Y_i| > \max\{(s-1)\cdot|X_i|, 2(s-1)\}$. Because the vertices in Y_i are not affected by the edge modifications in S, the fact that K_i is an s-plex implies

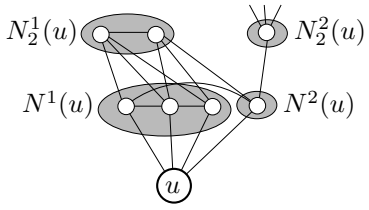

FIG. 3.1. *An illustration of the partitions of $N_2(u)$ and $N(u)$ for $\hat{s} = 2$. Vertex u satisfies the first two preconditions of Reduction Rule 2.*

that every vertex in X_i is adjacent to at least $|Y_i| - s + 1$ vertices in Y_i in the input graph G. With $|Y_i| > (s-1) \cdot |X_i|$, there has to be a vertex $u \in Y_i$ with $X_i \subseteq N_G(u)$ by the pigeonhole principle. Moreover, if $|Y_i| > 2(s-1)$, then the fact that K_i is an s-plex implies that $|Y_i \setminus N_G[u]| \leq s-1$, which means $|N_G[u] \cap Y_i| > s-1$. Thus, every vertex in Y_i has at least one neighbor in $N_G[u]$ and, hence, distance at most two to u in G.

Let us summarize our findings: If we do not apply a second data reduction rule to G, then there can be arbitrarily large s-plexes K_i in G_{plex}, in particular, $|Y_i| > \max\{(s-1) \cdot |X_i|, 2(s-1)\}$. However, then, there must be a vertex $u \in Y_i$ satisfying the following conditions:

C1. $X_i \subseteq N_G(u)$,
C2. $N_G[u] \subseteq K_i$,
C3. $|Y_i \setminus N_G[u]| \leq s-1$, and
C4. all vertices in $Y_i \setminus N_G[u]$ have distance two to u in G.

Thus, if $|Y_i|$ is very large, then $|N_G[u]|$ is very large and we need a data reduction rule to reduce $N_G[u]$. This is exactly what the second rule does.

To simplify notation, let $\hat{s} := s-1$ and write $N(u)$ and $N[u]$ for $N_G(u)$ and $N_G[u]$, respectively. Let $N_2(u)$ denote the set of vertices that have, in G, distance two to u. Further, we partition $N_2(u)$ into two sets, where the first set $N_2^1(u)$ consists of vertices *tightly coupled* with u:

$$N_2^1(u) := \{v \in N_2(u) : \ |N[u] \setminus N(v)| \leq \hat{s}\},$$
$$N_2^2(u) := N_2(u) \setminus N_2^1(u).$$

Analogously, $N(u)$ is also partitioned into two sets:

$$N^1(u) := \{v \in N(u) : \big(N(v) \subseteq N[u] \cup N_2^1(u)\big) \wedge \big(|N[v]| \geq |N[u] \cup N_2^1(u)| - \hat{s}\,\big)\},$$
$$N^2(u) := N(u) \setminus N^1(u).$$

Figure 3.1 illustrates the above definitions. It is easy to see that the sets $N^1(u)$, $N^2(u)$, $N_2^1(u)$, and $N_2^2(u)$ can be computed in $O(n^2)$ time for any vertex u.

Following the above analysis, we need a data reduction rule that shrinks the set of the vertices tightly coupled with a vertex u having a very special neighborhood (specified by conditions C1-C4). Using the newly introduced notation the situation can be described as follows: There can be many vertices in $N^1(u)$ but only few (at most $\hat{s}$) tightly coupled vertices in $N_2(u)$, that is, the $N_2^1(u)$-vertices. Moreover, these $N_2^1(u)$-vertices are only adjacent to vertices in $N(u)$ or to $N_2^1(u)$-vertices. Reduction Rule 2, which is shown in Figure 3.2, applies in this situation and replaces $N^1(u) \cup$

Reduction Rule 2:
If there is a vertex u for which
(1) $|N_2^1(u)| \leq \hat{s}$,
(2) $\forall v \in N_2^1(u) : (N(v) \subseteq N(u) \cup N_2^1(u)) \wedge (|N[v]| \geq |N[u] \cup N_2^1(u)| - \hat{s})$, and
(3) $|A| > \alpha+1$, where $A := \{u\} \cup N^1(u) \cup N_2^1(u)$ and $\alpha := 4\,\hat{s} \cdot (|N^2(u)| + |N_2^2(u)| + \hat{s})$,
then remove A from G and add a clique C to G with α vertices for even $|A|$ or $\alpha + 1$ vertices for odd $|A|$. To connect C to $G - A$, perform the following case distinction for every vertex $v \in N^2(u)$. Herein, for a vertex set U and a vertex $w \notin U$, let $U_w := U \cap N(w)$ and $\overline{U_w} := U \setminus U_w$.

Case 1. If $|A_v| - |\overline{A_v}| \geq |N^2(u)| + |N_2^2(u)|$, then connect v to $|C| - \min\{\hat{s}, |\overline{A_v}|\}$ many vertices of C and decrease the parameter k by $\max\{|\overline{A_v}| - \hat{s}, 0\}$.

Case 2. If $|\overline{A_v}| - |A_v| \geq |N^2(u)| + \hat{s}$, then decrease the parameter k by $|A_v|$.

Case 3. If $|N^2(u)| + |N_2^2(u)| > |A_v| - |\overline{A_v}| > -|N^2(u)| - \hat{s}$, then insert edges between v and the vertices in C such that $|C_v| - |\overline{C_v}| = |A_v| - |\overline{A_v}|$ and decrease the parameter k by $\max\{|A_v| - |C_v|, 0\}$.

FIG. 3.2. *Reduction Rule 2 of the kernelization algorithm.*

$N_2^1(u) \cup \{u\}$ by a smaller "simulating" clique. The basic idea behind Reduction Rule 2 is to consider a vertex u for which $A := \{u\} \cup N^1(u) \cup N_2^1(u)$ is an s-plex which only has edges to $N^2(u)$. The first two conditions of the rule ensure the s-plex property of A. If the third condition of Reduction Rule 2 is fulfilled, then A is too large, and thus it must be reduced. However, since the vertices in A may have distinct neighborhoods in $N^2(u)$, we cannot remove vertices from A arbitrarily (as it is possible for CLUSTER EDITING). The edges between a vertex $w \in N^2(u)$ and A determine whether w should be contained in the same s-plex as A. The idea is then to replace the whole set A by a smaller clique C and to connect C and $N^2(u)$ such that for every vertex w in $N^2(u)$ the edges between w and C "simulate" the connection between w and A, that is, w is together with C in an s-plex of the final cluster graph of the reduced instance if and only if w is with A in an s-plex of the final cluster graph of the original instance.

Before proving the correctness of Reduction Rule 2, we explain three points concerning its implementation which are important for proving the subsequent Lemma 3.2 and the correctness of this rule. As in Case 2 of Reduction Rule 2 no edges are inserted, the following discussion only addresses Cases 1 and 3. First, we fix an arbitrary vertex x in the clique C and make x adjacent to all vertices in $N^2(u)$ satisfying Cases 1 or 3. This is needed for guaranteeing that in the reduced graph G' there is a vertex $x \in C$ adjacent to all vertices in $N_{G'}(C)$ (note that $N_{G'}(C)$ is the set of vertices in $N^2(u)$ satisfying Cases 1 or 3). Second, when connecting a vertex v satisfying Cases 1 or 3 to C, we begin with a vertex in C which has the minimum degree in the current graph and proceed according to the non-descending ordering of the vertex degrees in C until $|C_v| = |C| - \min\{\hat{s}, |\overline{A_v}|\}$ (Case 1) or $|C_v| - |\overline{C_v}| = |A_v| - |\overline{A_v}|$ (Case 3). Finally, note that, since $|N^2(u)| + |N_2^2(u)| > |A_v| - |\overline{A_v}| > -|N^2(u)| - \hat{s}$ for every vertex v in $N^2(u)$ satisfying Case 3 and $|C| \geq \alpha$, it is always possible to connect v to some vertices in C such that $|C_v| - |\overline{C_v}| = |A_v| - |\overline{A_v}|$ with the described implementation.

To show the correctness of Reduction Rule 2, we need to prove that the input graph G has a solution S of size at most k if and only if the graph G' resulting from one application of this rule has a solution S' of size at most k', where k' is the new parameter after the application of the rule. To this end, we need two claims

(Lemma 3.1 and Lemma 3.2).

LEMMA 3.1. *Let u be a vertex satisfying the first two preconditions of Reduction Rule 2 and $|A| \geq \alpha$ with A and α defined as in Reduction Rule 2. Then, there exists an optimal solution generating an s-plex cluster graph which contains an s-plex K such that*

(a) $A \subseteq K$ and
(b) $K \subseteq A \cup N^2(u)$.

Proof. We first prove part (a). Let S_{plex} be an optimal solution for an input graph G that generates an s-plex cluster graph G_{plex}. If in G_{plex} the set $A = \{u\} \cup N^1(u) \cup N_2^1(u)$ is completely contained in one s-plex, then we are done; otherwise, we show that we can transform S_{plex} into a new optimal solution S satisfying (a). Let $K_1, \ldots, K_t$ be the s-plexes in G_{plex} with $K_i \cap A \neq \emptyset$. We define for $1 \leq i \leq t$

1. $\mathcal{A}_i := K_i \cap A$,
2. $\mathcal{B}_i := K_i \cap N^2(u)$ and
3. $\mathcal{C}_i := K_i \setminus (\mathcal{A}_i \cup \mathcal{B}_i)$.

We distinguish the following two cases.

First, if $|\mathcal{A}_i| + 2|\mathcal{B}_i| \leq |A| - \hat{s}$ for all $1 \leq i \leq t$, then we construct a set S of edge modifications that will generate a graph G' that, compared to G_{plex}, differs only in the K_i's, $1 \leq i \leq t$, and contains one more connected component $G[A]$, the subgraph of G induced by the vertex set A. More precisely, we remove all $\mathcal{A}_i$'s from the K_i's and merge them into one new connected component $G[A]$. All other s-plexes in G' remain the same as in G_{plex}. Let K_i' denote the corresponding modified K_i for $1 \leq i \leq t$. Thus, in G', each K_i' consists only of $\mathcal{B}_i$ and $\mathcal{C}_i$. Since every induced subgraph of an s-plex is clearly an s-plex, each K_i' is an s-plex. Consider the new component $G[A]$. By the precondition (1) of Reduction Rule 2, u has at most $\hat{s}$ non-adjacent vertices in $G[A]$. By the definition of $N^1(u)$, each vertex in $N^1(u)$ has at least $|A| - \hat{s}$ neighbors in $G[A]$. The same holds also for the vertices in $N_2^1(u)$ due to the precondition (2) of Reduction Rule 2. Thus, A is an s-plex and G' is an s-plex cluster graph. Therefore, S is also a solution set for G. Next, we prove that S is optimal. Compared with S_{plex}, the solution set S contains in addition to the edge modifications in S_{plex} the deletions of edges between $\mathcal{A}_i$ and $\mathcal{B}_i$ for $1 \leq i \leq t$, but saves the edge deletions between $\mathcal{A}_i$'s that are needed by S_{plex}. Note that graph G has no edge between $\mathcal{A}_i$ and $\mathcal{C}_i$, which follows from $\mathcal{C}_i \cap N[u] = \emptyset$, $\mathcal{A}_i \cap (N^2(u) \cup N_2^2(u)) = \emptyset$, and precondition (2) of Reduction Rule 2. On the one hand, the additional edge deletions in S amount to at most $\sum_{1 \leq i \leq t} |\mathcal{A}_i| \cdot |\mathcal{B}_i|$. On the other hand, the saved edge deletions amount to at least $(\sum_{1 \leq i \leq t} |\mathcal{A}_i| \cdot (|A| - |\mathcal{A}_i| - \hat{s}))/2$. By the precondition $|\mathcal{A}_i| + 2|\mathcal{B}_i| \leq |A| - \hat{s}$ for all $1 \leq i \leq t$, we have then $|S| \leq |S_{\text{plex}}|$ and S is optimal.

Second, if there is an s-plex K_j with $|\mathcal{A}_j| + 2|\mathcal{B}_j| > |A| - \hat{s}$, then we construct a set S of edge modifications that will generate a graph G' that, compared to G_{plex}, differs only in the K_i's. More precisely, we remove all $\mathcal{A}_i$'s from K_i's with $i \neq j$ and add these $\mathcal{A}_i$'s to K_j. Therefore, for $1 \leq i \leq t$ and $i \neq j$, the modified K_i's consist only of $\mathcal{B}_i$ and $\mathcal{C}_i$ in G'. Moreover, the modified K_j consists of $\mathcal{B}_j$, $\mathcal{C}_j$, and A. Obviously, the modified K_i's with $i \neq j$ remain s-plexes. However, by adding $\mathcal{A}_i$'s to $\mathcal{A}_j$, the modified K_j might not be an s-plex any more. On the one hand, in order to make the modified K_j an s-plex, we conduct the following additional edge modifications.

- Delete the edges between $\mathcal{A}_i$ and $\mathcal{B}_i$ for all $i \neq j$. These are at most $\sum_{i \neq j} |\mathcal{A}_i| \cdot |\mathcal{B}_i|$.
- Insert all missing edges between $\mathcal{A}_i$ and $\mathcal{C}_j$ for all $i \neq j$. These are at most $\sum_{i \neq j} |\mathcal{A}_i| \cdot |\mathcal{C}_j|$.

- Insert all missing edges between $\mathcal{A}_i$ and $\mathcal{A}_j \cup \mathcal{B}_j$ for all $i \neq j$. Note that $\mathcal{B}_j \subset N^2(u)$ and, thus, $\mathcal{A}_j \cup \mathcal{B}_j \subseteq N[u] \cup N_2^1(u)$. By precondition (1) of Reduction Rule 2, definition of $N^1(u)$, and precondition (2) of Reduction Rule 2, respectively, vertex u, each vertex in $N^1(u)$, each vertex in $N_2^1(u)$ have at most $\hat{s}$ non-adjacent vertices in $\mathcal{A}_j \cup \mathcal{B}_j$. Thus, we need at most $\sum_{i \neq j} |\mathcal{A}_i| \cdot \hat{s}$ edge insertions here.

In total we need at most

$$\sum_{i \neq j} |\mathcal{A}_i| \cdot (|\mathcal{B}_i| + |\mathcal{C}_j| + \hat{s}) \ \ (I)$$

additional edge modifications. On the other hand, however, we can undo the edge deletions between $\mathcal{A}_i$ and $\mathcal{A}_j \cup \mathcal{B}_j$ for all $i \neq j$, which amount to at least

$$\sum_{i \neq j} |\mathcal{A}_i| \cdot (|\mathcal{A}_j| + |\mathcal{B}_j| - \hat{s}) \ \ (II).$$

This is true because $\mathcal{A}_j \cup \mathcal{B}_j \subseteq N[u] \cup N_2^1(u)$ and $\mathcal{A}_i \subseteq \{u\} \cup N^1(u) \cup N_2^1(u)$. Next, we argue that the modified K_j is an s-plex. Then, we show that $(II) > (I)$ and, hence, the number of saved edge deletions is greater than the number of additional edge modifications, which implies that $|S| < |S_{\text{plex}}|$, contradicting the optimality of S_{plex}.

The fact that the modified K_j is an s-plex can be seen as follows. As analyzed in the first case, A and, thus, $\bigcup_{i \neq j} \mathcal{A}_i$ is an s-plex. Moreover, $\mathcal{A}_j \cup \mathcal{B}_j \cup \mathcal{C}_j$ (the original K_j) is also an s-plex. Hence, since we add all edges between $\bigcup_{i \neq j} \mathcal{A}_i$ and $\mathcal{A}_j \cup \mathcal{B}_j \cup \mathcal{C}_j$, every vertex remains non-adjacent to the same number of vertices, and hence the modified K_j is an s-plex.

For the proof that $(II) > (I)$, we first need the further observation that $\mathcal{C}_j \subseteq N_2^2(u)$. To this end, we show that $\mathcal{D}_j := \mathcal{C}_j \setminus N_2^2(u) = \emptyset$. Since K_j is an s-plex and $\mathcal{D}_j \cap (N(u) \cup N_2(u)) = \emptyset$, the optimal solution S_{plex} has to add at least $|\mathcal{A}_j| + |\mathcal{B}_j| - \hat{s}$ many edges to include a vertex in $\mathcal{D}_j$ into K_j. However, excluding all vertices in $\mathcal{D}_j$ from K_j needs at most $|\mathcal{C}_j'| \cdot |\mathcal{D}_j|$ edge deletions for $\mathcal{C}_j' := \mathcal{C}_j \cap N_2^2(u)$. Since $|A| > \alpha$ with $\alpha = 4\,\hat{s}(|N^2(u)| + |N_2^2(u)| + \hat{s})$, $\mathcal{B}_j \subseteq N^2(u)$, and $\mathcal{C}_j' \subseteq N_2^2(u)$, it follows that $|A| > |\mathcal{B}_j| + |\mathcal{C}_j'| + 2\,\hat{s}$. By $|\mathcal{A}_j| + 2|\mathcal{B}_j| > |A| - \hat{s}$ (the precondition of this case), we have $|\mathcal{A}_j| + |\mathcal{B}_j| > |\mathcal{C}_j'| + \hat{s}$. This means that only in the case $\mathcal{D}_j = \emptyset$, the edge modification set S is optimal and, thus, $\mathcal{C}_j \subseteq N_2^2(u)$.

To show that $(II) > (I)$, it remains to prove that

$$|\mathcal{A}_j| + |\mathcal{B}_j| - \hat{s} > |\mathcal{B}_i| + |\mathcal{C}_j| + \hat{s} \ \ (III).$$

By the case assumption we know that $|\mathcal{A}_j| + |\mathcal{B}_j| > |A| - |\mathcal{B}_j| - \hat{s}$ and, hence, we have $|\mathcal{A}_j| + |\mathcal{B}_j| - \hat{s} > |A| - |\mathcal{B}_j| - 2\,\hat{s}$. We show in the following that $|A| - |\mathcal{B}_j| - 2\,\hat{s} > |\mathcal{B}_i| + |\mathcal{C}_j| + \hat{s}$, implying (III). Note that the last inequality is equivalent to $|A| > |\mathcal{B}_i| + |\mathcal{B}_j| + |\mathcal{C}_j| + 3\,\hat{s}$. This inequality holds for the following reasons. First, by the precondition of this lemma, we have that $|A| \geq \alpha = 4\,\hat{s} \cdot (|N^2(u)| + |N_2^2(u)| + \hat{s})$. Second, we have that $4\,\hat{s} \cdot (|N^2(u)| + |N_2^2(u)| + \hat{s}) > |\mathcal{B}_i| + |\mathcal{B}_j| + |\mathcal{C}_j| + 2\,\hat{s}$ since $\mathcal{B}_i \cup \mathcal{B}_j \subseteq N^2(u)$ (by definition) and $\mathcal{C}_j \subseteq N_2^2(u)$ (by the above observation).

Part (b) of the lemma is easy to see. By part (a) of the lemma, there is an s-plex K with $A \subseteq K$. If $K \setminus (A \cup N^2(u)) \neq \emptyset$, then we construct a solution leading to an s-plex cluster graph where K is split into two s-plexes $K \cap (A \cup N^2(u))$ and $K \setminus (A \cup N^2(u))$. For every vertex $w \in K \setminus (A \cup N^2(u))$, removing w from K needs at most $|N^2(u)|$

edge deletions: w has no neighbor in A; however, adding w to K needs at least $|A| - \hat{s}$ edge insertions. Since $|A| \geq \alpha$, it is never better to include w into K. □

Now, before coming to the second claim (Lemma 3.2), we discuss in more detail the strategy behind the "simulating" clique C introduced in Reduction Rule 2. Based on Lemma 3.1, we can conclude that, with respect to a vertex u satisfying the preconditions of Reduction Rule 2, it remains to decide which vertices of $N^2(u)$ should form, together with A, an s-plex in the resulting s-plex cluster graph. Herein, Reduction Rule 2 distinguishes three cases. In the first two cases, a vertex $v \in N^2(u)$ has either much more or much less neighbors in A than outside of A (Cases 1 and 2). We can then easily decide whether v should be in the same s-plex with A (Case 1) or not (Case 2) and make the corresponding edge modifications. However, in the third case, where the "neighborhood size difference" is not so huge for a vertex $v \in N^2(u)$, the decision whether or not to put v into the same s-plex with A could be influenced by the global structure outside of $N(u) \cup N_2(u)$. To overcome this difficulty, Reduction Rule 2 makes use of the simulating clique C which is meant to play the same role as A but has a bounded size. Moreover, for every vertex $v \in N^2(u)$ the construction of C in Reduction Rule 2 guarantees that after its application v again adheres to the same case (Cases 1–3, distinguishing according to the neighborhood size difference of v) as it has before. Lemma 3.2 shows then that C plays the same role as A.

Lemma 3.2. *Let u be a vertex satisfying the three preconditions of Reduction Rule 2 and let G' denote the graph resulting from applying Reduction Rule 2 once to u. Then, in G', with the described implementation of Reduction Rule 2, each vertex in clique C is adjacent to all but at most $\hat{s}$ vertices in $N_{G'}(C)$.*

Proof. Observe that applying Reduction Rule 2 to G, Case 2 of the rule inserts no edge and, the vertices in $N^2(u)$, which have edges to C, have to satisfy either Case 1 or Case 3 of Reduction Rule 2. Let R_1 be the set of vertices in $N_{G'}(C)$ satisfying Case 1 in G and let R_2 be the set of vertices in $N_{G'}(C)$ satisfying Case 3 in G, giving $N_{G'}(C) = R_1 \cup R_2$.

First, consider the case $|R_2| \geq 2\,\hat{s}$. Consider a vertex $v \in R_1$. In G, v has $|A_v|$ many edges to A with $A_v = A \cap N_G(v)$. In contrast, v has in G' only $C - \min\{\hat{s}, |\overline{A_v}|\}$ many edges to C. This means that v loses at most $|A| - |C|$ incident edges in G'. A vertex $v \in R_2$ loses exactly $(|A| - |C|)/2$ incident edges from G to G'. According to the definition of A and the fact that $N_{G'}(C) \subseteq N^2(u)$, G' has at least

$$(|R_1| + |R_2| - \hat{s}) \cdot |A| - |R_1| \cdot (|A| - |C|) - |R_2| \cdot (|A| - |C|)/2$$

edges between $R_1 \cup R_2$ and C. This number equals

$$|A| \cdot (|R_2|/2 - \hat{s}) + |C| \cdot (|R_2|/2 + |R_1|),$$

and since $|R_2| \geq 2\,\hat{s}$ and $|A| \geq |C|$, the number of edges between C and $R_1 \cup R_2$ is at least $|C| \cdot (|R_1| + |R_2| - \hat{s})$. This implies that the above implementation of Reduction Rule 2, where a vertex in C with the minimum degree is preferred while connecting vertices in $R_1 \cup R_2$ to C, always results in a graph G' where every vertex in C has at least $|R_1| + |R_2| - \hat{s}$ neighbors in $N_{G'}(C)$.

Second, consider the case $|R_2| < 2\,\hat{s}$. For every vertex $v \in R_2$ by the precondition $|\overline{A_v}| - |A_v| < |N^2(u)| + \hat{s}$ of Case 3, $|\overline{C_v}| - |C_v| = |\overline{A_v}| - |A_v|$, and $|C| = |\overline{C_v}| + |C_v|$, we have

$$|C_v| > \frac{|C| - |N^2(u)| - \hat{s}}{2}.$$

Observe that if $|C_v| \geq |C|/2$, after adding the edges between the vertices in R_2 and C according to the described implementation, then, with $|R_2| < 2\,\hat{s}$, at least $|C|/2$ vertices of C have at least $|R_2| - \hat{s} + 1$ neighbors in R_2. Thus, with $|C_v| > |C|/2 - (N^2(u) + \hat{s})/2$ and $2\,\hat{s} \cdot (|N^2(u)| + \hat{s})/2 < |C|/2$ (following from $|C| \geq \alpha$), we can conclude that, after adding the edges between the vertices in R_2 and C according to the described implementation, there are at most $|C|/2 + \hat{s} \cdot (|N^2(u)| + \hat{s})$ vertices in C adjacent to $|R_2| - \hat{s}$ vertices in R_2. The other vertices in C are adjacent to $|R_2| - \hat{s} + 1$ vertices in R_2. Now consider the vertices in R_1. By Case 1 ($|C_v| \geq |C| - \hat{s}$) and the described implementation, there are at most $|C|/2 + \hat{s} \cdot (|N^2(u)| + \hat{s}) + |R_1| \cdot \hat{s}$ vertices in C adjacent to $|R_1 \cup R_2| - \hat{s}$ vertices in $R_1 \cup R_2$. The other vertices in C have $|R_1 \cup R_2| - \hat{s} + 1$ neighbors in $R_1 \cup R_2$. It is easy to see that $\hat{s} \cdot (|N^2(u)| + \hat{s}) + |R_1| \cdot \hat{s} < |C|/2$ and, thus, every vertex in C has at most $\hat{s}$ non-adjacent vertices in $R_1 \cup R_2 = N_{G'}(C)$. □

With Lemma 3.1 and Lemma 3.2, we can prove the correctness and the running time of Reduction Rule 2.

LEMMA 3.3. *Reduction Rule 2 is correct and can be carried out in $O(n^3)$ time.*

Proof. Let u be a vertex satisfying the three preconditions of Reduction Rule 2. We prove its correctness by showing that the input graph G has a solution S with $|S| \leq k$ if and only if the graph G' after one application of Reduction Rule 2 to u has a solution S' with $|S'| \leq k'$, where k' is the new parameter value after the application.

"⇒": Due to Lemma 3.1, there is an optimal solution S for G which generates an s-plex cluster graph H with an s-plex K with $A \subseteq K$ and $K \subseteq A \cup N^2(u)$. Let $L := K \setminus A$. We will show in the following how to construct S' from S. The graph generated by S' will be called H'. The basic idea is to construct S' such that H' differs from H only in K. That is, with the exception of K, graph H' contains the same connected components as H. In addition, H' contains a connected component K' with the vertex set $L \cup C$. Hence, in the following we specify only how S' modifies the edges between vertices in K'.

Since G and G' differ only in A and C, it follows that S and S' differ only in the modifications of the edges incident to A or C. Observe that u, each vertex in $N^1(u)$, and each vertex in $N_2^1(u)$ have at most $\hat{s}$ non-adjacent vertices in $N[u] \cup N_2^1(u)$ due to precondition (1) of Reduction Rule 2, the definition of $N^1(u)$, and precondition (2) of Reduction Rule 2, respectively. From part (2) of Lemma 3.1 it follows that S does not modify edges between two vertices in A. Correspondingly, S' modifies no edge in C. Thus, we only have to compare the number of modifications of the edges incident to exactly one vertex in A or C, that is, the edges between A and $N^2(u)$ or between C and $N^2(u)$. We distinguish four cases for the vertices $v \in N^2(u)$ and show that, in each of these cases, the difference between the number of edge modifications made by S and the number of edge modifications made by S' is equal to the decrement of the parameter k caused by Reduction Rule 2.

First, assume that v satisfies Case 1 of Reduction Rule 2. We can conclude that $v \in K$: To separate v from A, one would need to delete at least $|A_v|$ edges. However, to include v in K requires at most $|\overline{A_v}|$ edge insertions (connecting v to all vertices in A), $|N_2^2(u)|$ edge deletions (disconnecting v and $N_2^2(u)$), and $|N^2(u)|$ edge deletions (disconnecting v and $K \setminus N^2(u)$) and insertions (connecting v and $K \cap N^2(u)$). Due to the precondition of Case 1, we have $v \in K$ and S inserts $|\overline{A_v}| - |X|$ edges between v and A with X being the set of vertices in A that are not adjacent to v in H; clearly, $|X| \leq \hat{s}$. In H', S' inserts $|\overline{C_v}| - |X|$ many edges between v and C. According to Case 1 of Reduction Rule 2, $|\overline{C_v}| = \min\{\hat{s}, |\overline{A_v}|\}$. Thus, $|S| - |S'|$ with respect to

the edges between v and C is equal to $|\overline{A_v}| - |X| - |\overline{C_v}| + |X| = \max\{|\overline{A_v}| - \hat{s}, 0\}$, the same as the decrement of the parameter caused by Case 1 of Reduction Rule 2. Finally, we can easily observe that there are at most $\hat{s}$ vertices in K' not adjacent to v.

Second, assume that v satisfies Case 2 of Reduction Rule 2. Observe that excluding v from K needs at most $|A_v| + |N^2(u)| + |N_2^2(u)|$ edge modifications, while including v in K requires at least $|\overline{A_v}| - \hat{s} + |N_2^2(u)|$ edge modifications. Thus, $v \notin K$ and S deletes $|A_v|$ edges between v and A. Clearly, $v \notin K'$ in G'. Since G' has no edge between v and C, S' modifies no edge between v and C. So, the difference between $|S|$ and $|S'|$ with respect to the edges between v and C is $|A_v|$, the same as the decrement of the parameter in Case 2 of Reduction Rule 2.

Third, assume that v satisfies Case 3 of Reduction Rule 2 and $v \notin K$. Then, $v \notin K'$. Thus, S deletes $|A_v|$ edges between v and A, while S' deletes $|C_v|$ edges between v and C. The difference between the number of edge deletions applied to v in S and S' is thus $|A_v| - |C_v|$, exactly the decrement of the parameter in Case 3 of Reduction Rule 2.

Finally, assume that v satisfies Case 3 and $v \in K$. Suppose that S inserts $|\overline{A_v}| - |X|$ edges between v and A, where X denotes the set of vertices in A that are not adjacent to v in H. In H', S' inserts $|\overline{C_v}| - |X|$ edges between v and C. Thus, the difference is then $|\overline{A_v}| - |\overline{C_v}| = |A_v| - |C_v|$, exactly the decrement of the parameter in Case 3 of Reduction Rule 2. Clearly, v has at most $\hat{s}$ non-adjacent vertices in K if and only if v has at most $\hat{s}$ non-adjacent vertices in K'.

Summarizing these four cases, we can conclude $|S'| \leq k'$. It remains to show that H' is an s-plex cluster graph. Since H and H' differ only in the connected components K and K', we only have to show that K' is an s-plex. As argued in the first and last cases of the above four cases, all vertices in $K' \cap N^2(u)$ have at most $\hat{s}$ non-adjacent vertices in K'. By Lemma 3.2, the vertices in the clique C have degree at least $|C| + |N_{G'}(C)| - \hat{s}$ in G'. Since, according to the construction of H', $K' = C \cup L$ with $L = K \setminus A$ and $K \subseteq A \cup N^2(u)$, we can conclude that K' is an s-plex and H' is an s-plex cluster graph. Together with $|S'| \leq k'$, the "$\Rightarrow$"-direction follows.

"$\Leftarrow$": The first step of the proof of this direction is to show that there is a solution S' for G' that generates an s-plex K' with $C \subseteq K' \subseteq C \cup N_{G'}(C)$. To this end, it suffices to show that there exists a vertex $u' \in C$ such that the preconditions of Reduction Rule 2 are fulfilled by u'. Then, the claim follows from Lemma 3.1. From the implementation of Reduction Rule 2, we know that there is a vertex x in C adjacent to all vertices in $N_{G'}(C)$. We now show that x fulfills the preconditions of Reduction Rule 2: Clearly, $N_{G'}[x] = C \cup N_{G'}(C)$. Let $N_{G',2}(x)$ denote the set of vertices having in G' distance two to x. Since all vertices in C have only neighbors in $N_{G'}[x]$ and $|C| > \hat{s}$, no vertex $v \in N_{G',2}(x)$ satisfies $|N_{G'}(v) \setminus N_{G'}(x)| \leq \hat{s}$ and, thus, $N^1_{G',2}(x) = \emptyset$. By Lemma 3.2, every vertex in C has at least $|N_{G'}(C)| - \hat{s}$ neighbors in $N_{G'}(C)$ and, thus, $C \subseteq N^1_{G'}(x)$. Therefore, x fulfills preconditions (1) and (2) of Reduction Rule 2. Furthermore, $N_{G',2}(x) \subseteq N^2(u) \cup N_2^2(u)$. It is easy to observe that $N^2_{G'}(x) \cup N^2_{G',2}(x) \subseteq N^2(u) \cup N_2^2(u)$. Since $|C| \geq \alpha$, x fulfills all preconditions of Lemma 3.1. The remaining part of the proof follows almost the same principle as the proof of the "$\Rightarrow$"-direction, that is, constructing a graph H from the s-plex cluster graph H' resulting from applying a solution S' to G', and then proving that the set of the edge modifications needed to transform G into H has size at most k and that H is an s-plex cluster graph. The one-to-one correspondence between the two solutions can be proven in the same way as the proof for the opposite direction.

Concerning the running time of Reduction Rule 2, note that we iterate over all vertices to check the applicability of the rule. For each vertex u, we compute $N^1(u)$, $N^2(u)$, $N_1^1(u)$, and $N_2^2(u)$ in $O(n^2)$ time, and connecting the vertices in $N^2(u)$ to C requires $O(n^2)$ time. Altogether, the application of Reduction Rule 2 needs $O(n^3)$ time. □

Finally, we prove the main theorem in this section.

THEOREM 3.4. *s*-PLEX CLUSTER EDITING *admits a* $(8s^2-6)\cdot k+8(s-1)^2$*-vertex problem kernel for* $s \geq 2$*. It can be computed in* $O(n^4)$ *time.*

Proof. Let G_{plex} denote the s-plex cluster graph resulting from applying a solution S with $|S| \leq k$ to the input graph $G = (V, E)$, and let $K_1, \ldots, K_l$ be the s-plexes in G_{plex}. The set V of vertices of G_{plex} can be partitioned into two subsets, namely, X, the set of vertices that are endpoints of the edges modified by S, and $Y := V \setminus X$. For an s-plex K_i, let $X_i := X \cap K_i$ and $Y_i := Y \cap K_i$. As stated in the beginning of this section, we know that $|X| \leq 2k$. Moreover, if $|Y_i| > \max\{\hat{s} \cdot |X_i|, 2\,\hat{s}\}$ for some i, then there must be a vertex $u \in Y_i$ that satisfies C1-C4 given there. Since, for each vertex $v \in N_2^1(u)$, $|N[v] \setminus N(u)| \leq s-1$ holds, v cannot be affected by S and, thus, $v \in Y_i$. This means $|N_2^1(u)| \leq s-1$ (precondition (1) of Reduction Rule 2). Furthermore, since K_i is an s-plex and $N[u] \subseteq K_i$, all vertices $v \in N_2^1(u)$ should satisfy $N(v) \subseteq N(u) \cup N_2^1(u)$ and $|N[v]| \geq |N[u] \cup N_2^1(u)| - s + 1$ (precondition (2) of Reduction Rule 2). Clearly, all vertices in Y_i are from $N[u] \cup N_2(u)$ and none of them can be in $N^2(u) \cup N_2^2(u)$; otherwise, there would be an edge modification affecting them. Since $Y_i \subseteq N^1(u) \cup N_2^1(u) \cup \{u\}$, this implies either $|Y_i| \leq \alpha := 4\,\hat{s} \cdot (|N^2(u)| + |N_2^2(u)| + \hat{s})$ or Reduction Rule 2 can be applied to u. If we assume that the input graph is reduced with respect to both data reduction rules, then the former case applies. Clearly, $X_i \subseteq N^2(u)$. The vertices in $N_2^2(u)$ are in X as well, but they are not in X_i, which means that we cannot give an upper bound for $|Y_i|$ only depending on $|X_i|$. However, if we consider all Y_i's together, then, for every modified edge, each of its two endpoints in X might be counted twice, once in $N^2(v)$ for a vertex $v \in K_i \cap Y$ and once in $N_2^2(w)$ for another vertex $w \in K_j \cap Y$ with $i \neq j$. Hence, considering all K_i's we then have

$$\begin{aligned}\sum_{1 \leq i \leq l} |Y_i| &\leq \sum_{1 \leq i \leq l} \max\{2\,\hat{s}, \hat{s} \cdot |X_i|, 8\,\hat{s} \cdot (|X_i| + \hat{s}))\} \\ &\overset{(***)}{\leq} 16\,\hat{s}\,k + 8\,\hat{s}^2 \cdot (k+1).\end{aligned}$$

Inequality (***) follows from $|X| \leq 2k$ and the fact that deleting at most k edges from a connected graph results in at most $k+1$ connected components. Together with $|X| \leq 2k$, we obtain a problem kernel with $|X| + |Y| \leq 8\,\hat{s}^2(k+1) + 16\,\hat{s}\,k + 2k = (8s^2-6)k + 8(s-1)^2$ vertices.

The running time $O(n^4)$ follows directly from Lemma 3.3 and the fact that Reduction Rule 2 can be applied at most n times. □

The kernelization algorithm of Theorem 3.4 can be adapted to yield a $6k$-vertex problem kernel for $s = 1$, that is, CLUSTER EDITING. We omit the details since this does not improve on the currently best $2k$-vertex problem kernel for CLUSTER EDITING due to Chen and Meng [7].

4. Forbidden Subgraph Characterization. In this section, we present a characterization of s-plex cluster graphs by means of forbidden *induced* subgraphs for any $s \geq 1$. More specifically, we provide a set $\mathcal{F}$ of graphs such that a graph G is an s-plex cluster graph if and only if G is $\mathcal{F}$*-free*, that is, G does not contain any

induced subgraph from $\mathcal{F}$. If $s = 1$, where all connected components of the cluster graph are required to form cliques, the only forbidden induced subgraph is a path on three vertices [32]. In contrast, if $s \geq 2$, we face up to exponentially in s many forbidden induced subgraphs. To cope with this, we develop a characterization of these subgraphs that still allows us to derive efficient algorithms. More specifically, in Section 4.1, we show that s-plex cluster graphs are characterized by forbidden induced subgraphs with $O(s)$ vertices. Moreover, in Section 4.2, we show that, given a graph that is not an s-plex cluster graph, a forbidden induced subgraph can be found in $O(s \cdot (n+m))$ time. Based on this, in Section 5 we will present a branching strategy for s-Plex Cluster Editing that leads to a search tree of size $O((2s + \lfloor\sqrt{s}\rfloor)^k)$.

4.1. Description of the Forbidden Induced Subgraphs. A graph H is a *minimal forbidden induced subgraph* if it is not an s-plex cluster graph but every induced proper subgraph of H is an s-plex cluster graph. In the following, our goal is to identify and describe the set $\mathcal{F}_{s,\min}$ of all minimal forbidden induced subgraphs for s-plex cluster graphs. More precisely, we first give a graph-theoretic description of the minimal forbidden induced subgraphs. Then, we show that the number of vertices in every minimal forbidden induced subgraph is upper-bounded by $s + t_s + 1$, where

$$t_s := \lfloor -0.5 + \sqrt{0.25 + s} \rfloor$$

(note that for a non-negative integer i it holds that $i \cdot (i+1) \leq s$ if and only if $i \leq t_s$). Finally, we show that the upper bound of $s + t_s + 1$ on the number of vertices of a minimal forbidden induced subgraph is tight. That is, we show that for every $s \geq 2$ there exist minimal forbidden induced subgraphs with exactly this number of vertices.

We begin with a graph-theoretic description of the minimal forbidden induced subgraphs. The starting point are the connected graphs that contain a vertex that is not adjacent to exactly s other vertices. These graphs clearly are not s-plex cluster graphs. Let $\mathcal{C}$ denote the set of connected graphs. Define

$$\mathcal{C}(s,i) := \{H = (W,F) \in \mathcal{C} \mid (|W| = s + i + 1) \wedge (\exists w \in W : \deg_H(w) = i)\}$$

and

$$\mathcal{F}(s,j) := \bigcup_{i=1}^{j} \mathcal{C}(s,i).$$

Next, motivating the definitions of $\mathcal{C}(s,i)$ and $\mathcal{F}(s,j)$, we show that all minimal forbidden induced subgraphs are contained in $\mathcal{F}(s, n-s-1)$. Then, we refine this characterization by showing that a graph from $\mathcal{C}(s,i)$ is minimal if and only if its minimum vertex degree is i and all neighbors of a degree-i vertex are cut-vertices.

Lemma 4.1. *G is an s-plex cluster graph if and only if G is $\mathcal{F}(s, n-s-1)$-free.*

Proof. "$\Rightarrow$": Since the property of being an s-plex cluster graph is hereditary, all induced subgraphs of G are s-plex cluster graphs. Hence, G is $\mathcal{F}(s, n-s-1)$-free since the graphs in $\mathcal{F}(s, n-s-1)$ are not s-plex cluster graphs.

"$\Leftarrow$": We show the contraposition. If G is not an s-plex cluster graph, then G contains a connected component $C = (W, F)$ that is not an s-plex and $|W| \geq s + 2$ (note that every connected component with at most $s + 1$ vertices is an s-plex). Consider a vertex $v \in W$ of minimum degree. Since W does not form an s-plex, it contains at least s vertices not adjacent to v. Hence, we can find an induced subgraph of G from $\mathcal{C}(s, \deg_G(v)) \subseteq \mathcal{F}(s, n-s-1)$ using breadth-first search starting at v. □

Next, we precisely describe the minimal forbidden induced subgraphs.

LEMMA 4.2. *A graph $H \in \mathcal{C}(s,i)$ is a minimal forbidden induced subgraph if and only if the minimum vertex degree of H is i and for every $v \in V(H)$ with $\deg_H(v) = i$ the neighbors of v are cut-vertices.*

Proof. "$\Rightarrow$": We show the contraposition. First, assume that H contains a vertex v with $\deg_H(v) < i$. Then, by breadth-first search starting at v, we can find a set $S \subset V(H)$ such that $H[S] \in \mathcal{C}(s, \deg(v))$. Hence, H is not minimal. Second, assume that there exists a degree-i vertex v with a neighbor u that is not a cut-vertex. Then, by deleting u we obtain a graph from $\mathcal{C}(s, i-1)$, that is, H is also not minimal. "$\Leftarrow$": Consider an arbitrary vertex $v \in V(H)$. We show that $H' := H - v$ is an s-plex cluster graph. Note that H' contains $s+i$ vertices and the minimum vertex degree in H' is $i-1$. First, all vertices $w \in V(H')$ with $\deg_{H'}(w) \geq i$ are not adjacent to at most $s-1$ other vertices. Second, consider an arbitrary degree-$(i-1)$ vertex $u \in V(H')$. Note that u is a neighbor of v in H (since the minimum degree in H is i). Therefore, v is a cut-vertex in H and the deletion of v separates from u at least one of the s vertices non-adjacent to u in H. As a consequence, u is not adjacent to at most $s-1$ other vertices in the connected component of H' containing u. In summary, every vertex $w \in V(H')$ is not adjacent to at most $s-1$ other vertices in the connected component in H' in which it is contained. Hence, H' is an s-plex cluster graph. □

Based on the description of the minimal forbidden induced subgraphs given in Lemma 4.2, we show that the number of vertices in a minimal forbidden induced subgraph is bounded by $O(s)$. To this end, we show that the set $\mathcal{F}_{s,\min}$ of all minimal forbidden induced subgraphs is contained in $\mathcal{F}(s,t_s)$, that is, the number of vertices of every minimal forbidden induced subgraph is upper-bounded by $s+t_s+1$.

THEOREM 4.3. *A graph is an s-plex cluster graph if and only if it is $\mathcal{F}(s,t_s)$-free.*

Proof. It is sufficient to show that all minimal forbidden induced subgraphs are contained in $\mathcal{F}(s,t_s)$. Assume towards a contradiction that there exists a minimal forbidden induced subgraph H not contained in $\mathcal{F}(s,t_s)$. By Lemma 4.1, $H \in \mathcal{F}(s, n-s-1)$. This implies that $H \in \mathcal{C}(s,i)$ for some i with $i \cdot (i+1) > s$ (or, equivalently, $i > t_s$). Since H is a minimal forbidden induced subgraph, according to Lemma 4.2, the minimum vertex degree of H is i. Let $v \in V(H)$ be a degree-i vertex. Note that, according to Lemma 4.2, all neighboring vertices $N_H(v) = \{u_1, u_2, \ldots, u_i\}$ of v are cut-vertices. For every neighboring vertex u_j of v let U_j denote the set of vertices in $V(H)$ that are not reachable from v in $H - u_j$. On the one hand, note that $|U_j| \geq i$ for every $1 \leq j \leq i$ since the minimum vertex degree of H is i and since for every vertex $w \in U_j$ it holds that $N_H(w) \subseteq (U_j \cup \{u_j\}) \setminus \{w\}$. On the other hand, since v has degree $i = |V(H)| - s - 1$ in H, we have $\sum_{j=1}^{i} |U_j| \leq s$. Hence, there must exist at least one r, $1 \leq r \leq i$, with $|U_r| \leq s/i < i \cdot (i+1)/i = i+1$. Therefore, $|U_r| = i$. Moreover, since the minimum vertex degree in H is i, $U_r \cup \{u_r\}$ forms a clique of size $i+1$ and thus by deleting all but one vertex of U_r we obtain a graph in $\mathcal{C}(s,1)$ which is by definition not an s-plex cluster graph. This contradicts the fact that H is a minimal forbidden induced subgraph. □

So far, we have shown that the number of vertices of every minimal forbidden induced subgraph is bounded from above by $s+t_s+1$. Clearly, this implies that the number of minimal forbidden induced subgraphs is bounded from above by a function of s. The number of minimal forbidden induced subgraphs may, however, be exponential in s. Note that the maximum number of vertices in a minimal forbidden induced subgraph is of greater algorithmic importance than the exact number of

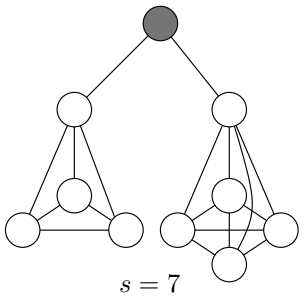

FIG. 4.1. *A minimal forbidden induced subgraph on 10 vertices for 7-plex cluster graphs. The gray vertex is a vertex with minimum degree and all its neighbors are cut-vertices.*

the minimal forbidden induced subgraphs: providing a smaller upper bound would directly improve the running time of the search tree algorithms presented in Section 5. However, we show that the upper bound of $s + t_s + 1$ on the number of vertices in a minimal forbidden induced subgraph is tight. To this end, we show that for every $s \geq 2$ a minimal forbidden induced subgraph is contained in $\mathcal{C}(s, t_s)$.

Consider the graph shown in Figure 4.1. It contains 10 vertices and the gray vertex has degree two. Hence, this graph is contained in $\mathcal{C}(7, 2)$ (note that $t_7 = 2$). Moreover, the gray vertex is the only vertex with degree two and all its neighbors are cut-vertices. Hence, according to Lemma 4.2 the shown graph is a minimal forbidden induced subgraph for $s = 7$. This example can be generalized to every $s \geq 2$. That is, for every s we can construct a minimal forbidden induced subgraph contained in $\mathcal{C}(s, t_s)$ as follows. We start with a star with center vertex v and t_s leaves, say $u_1, \ldots, u_{t_s}$. Then, for every u_j we add a clique C_j in such a way that $\sum_{j=1}^{t_s} |C_j| = s$ and that all cliques are of same size plus/minus one. Finally, we make all vertices in C_j adjacent to u_j, $1 \leq j \leq t_s$. Since $\sum_{j=1}^{t_s} |C_j| = s$ (and $t_s \cdot (t_s + 1) \leq s$ by definition), we have that $|C_j| \geq t_s + 1$, and, as a consequence, v is the only vertex of degree t_s and the degree of all other vertices is at least $t_s + 1$. Since the deletion of u_j separates the vertices in C_j from v, all neighbors of v are cut-vertices. Hence, by Lemma 4.2, the constructed graph is a minimal forbidden induced subgraph.

Summarizing, we arrive at the following.

PROPOSITION 4.4. *For every $s \geq 2$ there exists a minimal forbidden induced subgraph contained in $\mathcal{C}(s, t_s)$.*

4.2. Finding a Minimal Forbidden Induced Subgraph. In this subsection, we focus on efficiently finding a minimal forbidden induced subgraph. We show that, given a graph G that is not an s-plex cluster graph, a minimal forbidden induced subgraph can be found in $O(s \cdot (n + m))$ time.

Since the number of vertices in every minimal forbidden induced subgraph is bounded by a function in s, the number of minimal forbidden induced subgraphs is bounded by a function in s as well (which is exponential). A simple enumerative approach to detect a forbidden induced subgraph fails due to the sheer combinatorial explosion. Therefore, we first present an algorithm (Algorithm A, see Figure 4.2) that, given a graph $H = (W, F) \in \mathcal{C}(s, i)$ (for some $i \geq 1$), checks in $O(|W| + |F|)$ time whether H is a minimal forbidden induced subgraph. If this is the case, then it outputs "yes", otherwise it returns an induced subgraph of H contained in $\mathcal{C}(s, i')$ with $i' < i$. Clearly, we can use Algorithm A iteratively to find a minimal forbidden induced subgraph. However, when starting with an arbitrary graph contained in $\mathcal{F}(s, n-s-1)$,

Input: $H = (W, F)$ from $\mathcal{C}(s, i)$.
Output: "yes", if H is a minimal forbidden subgraph;
an induced subgraph $H' \in \mathcal{C}(s, i')$ of H with $i' < i$, otherwise.

1 $v := \arg\min_{w \in W}\{\deg_H(w)\}$
2 **if** $\deg_H(v) < i$ **then**
3 **return** a connected graph induced by $N_H[v] \cup S$
where S contains s vertices from $W \setminus N_H[v]$
4 Let Cutvertices be the set of cut-vertices of H
5 **if** $\exists v \in W : (N_H(v) \setminus \text{Cutvertices}) \neq \emptyset$ **then**
6 **return** graph $H - w$ for an arbitrary $w \in (N_H(v) \setminus \text{Cutvertices})$
7 **return** "yes"

FIG. 4.2. *Algorithm A checks whether a forbidden induced subgraph is minimal. If not, it computes a smaller forbidden induced subgraph.*

Algorithm A has to be applied up to n times in the worst case, resulting in an overall running time of $O(n \cdot m)$ for finding a minimal forbidden induced subgraph. To achieve linear running time, we develop an algorithm (Algorithm B, see Figure 4.3) that, given a forbidden induced subgraph $H = (W, F)$ from $\mathcal{F}(s, n - s + 1)$, finds in $O(s \cdot (|W| + |F|))$ time a forbidden induced subgraph from $\mathcal{F}(s, s)$. Since the number of vertices in such a subgraph is upper-bounded by $O(s)$, we can then apply Algorithm A iteratively $O(s)$ times to obtain a minimal forbidden induced subgraph.

Next, we present Algorithm A (see Figure 4.2) and prove its correctness.

LEMMA 4.5. *Let $H = (W, F) \in \mathcal{C}(s, i)$. Algorithm A (Figure 4.2) computes in $O(|W| + |F|)$ time an induced subgraph $H' \in \mathcal{C}(s, i')$ of H, with $i' < i$, or outputs "yes" if H is a minimal forbidden induced subgraph.*

Proof. Consider lines 1 to 3 of the algorithm. If a vertex v in H has degree less than i, then we can clearly find a graph from $\mathcal{C}(s, \deg_H(v))$ by choosing $N_H[v]$ and a set $S \subseteq W \setminus N_H[v]$ of s further (arbitrary) vertices such that $H[N_H[v] \cup S]$ is connected. This is doable in linear time by breadth-first search starting at v.

Consider lines 4 to 6. If one of the neighboring vertices of v, say w, is no cut-vertex, then we can delete w from H obtaining a graph from $\mathcal{C}(s, i - 1)$. Note that cut-vertices can be computed in linear time [33].

Consider line 7. The minimum vertex degree of H is i and the neighbors of every degree-i vertex are cut-vertices. Hence, according to Lemma 4.2, H is a minimal forbidden induced subgraph. □

Next, we present Algorithm B (see Figure 4.3) that, given a large forbidden induced subgraph, finds a forbidden induced subgraph with $O(s)$ vertices.

LEMMA 4.6. *Let $H = (W, F) \in \mathcal{C}(s, i)$ with $i > s$. Algorithm B (Figure 4.3) computes in $O(s \cdot (|W| + |F|))$ time an induced subgraph $H' \in \mathcal{C}(s, i')$ of H with $i' \leq s$.*

Proof. Consider lines 1 to 3. If $\deg_H(u) \leq s$, then we can clearly find a set $S \subset W \setminus N_H[u]$ of s vertices such that $H[N_H[u] \cup S]$ is connected and $|S| = s$. This graph is in $\mathcal{C}(s, i')$ for some $i' \leq s$.

In the following, let v denote a vertex with degree i (line 4). We use the following observation:

If one of the neighboring vertices of a degree-i vertex v is a cut-vertex,
then there exists at least one vertex in H with degree at most s.

This can be seen as follows. Assume that $x \in N_H(v)$ is a cut-vertex and let $U \subseteq W$ denote the vertices not reachable from v in $H - x$. Since a vertex $w \in U$ can

Input: $H = (W, F)$ from $\mathcal{C}(s, i)$ with $i > s$
Output: An induced subgraph $H' \in \mathcal{C}(s, i')$ of H with $i' \leq s$
1 $u := \arg\min_{w \in V}\{\deg_H(w)\}$
2 **if** $\deg_H(u) \leq s$ **then**
3 **return** a connected graph induced by $N_H[u] \cup S$ where S contains s vertices from $W \setminus N_H[u]$
4 Let $v \in V$ be a vertex with $\deg_H(v) = i$
5 $N_H(v) := \{u_1, u_2, \ldots, u_i\}$
6 Let $K := \{K_1, K_2, \ldots, K_l\}$ with $l \leq s$ denote the connected components of $H - N_H[v]$
7 Construct an auxiliary bipartite graph $B = (X_N, X_K, R)$ with
8 $X_N := \{x_{u_j} \mid 1 \leq j \leq i\}$,
9 $X_K := \{x_{K_q} \mid 1 \leq q \leq l\}$, and
10 $R := \{\{x_{u_j}, x_{K_q}\} \mid \exists\{u_j, v'\} \in F \text{ with } v' \in K_q\}$
11 $r := \arg\min_{q \in \{1,\ldots,l\}}\{\deg_B(x_{K_q})\}$
12 $CC := \{u_j \mid x_{u_j} \in N_B(x_{K_r})\}$
13 $\hat{H} := H - (CC \setminus \{w\})$ for an arbitrary vertex $w \in CC$
14 $v' := \arg\min_{w \in V(\hat{H})}\{\deg_{\hat{H}}(w)\}$
15 **return** a connected graph induced by $N_{\hat{H}}[v'] \cup S$ where S contains s vertices from $V(\hat{H} \setminus N_{\hat{H}}[v']$

FIG. 4.3. *Algorithm B to compute a forbidden induced subgraph with $O(s)$ vertices.*

only be adjacent to vertices in $U \cup \{x\}$ and $|U| \leq s$ (by definition of $\mathcal{C}(s, i)$), we have $\deg_H(w) \leq s$.

According to this observation, when entering line 5 of Algorithm B, we know that none of the vertices in $N_H(v) = \{u_1, u_2, \ldots, u_i\}$ is a cut-vertex. To make use of the observation, the remaining part of the algorithm is devoted to finding a set of vertices from $N_H(v)$ whose removal leads to a connected graph in which one neighbor of v is a cut-vertex. To this end, one constructs an auxiliary bipartite graph $B = (X_N, X_K, R)$ (lines 5-10). Concerning the running time needed for the construction of B, note that the degree of a vertex in X_N is at most s since $H - N_H[v]$ contains exactly s vertices and, hence, X_K has size at most s. Thus, to define R, one iterates over the edge set F and, given an edge $\{u_j, v'\}$ with $v' \in K_q$, one can decide in $O(s)$ time whether the edge $\{x_{u_j}, x_{K_q}\}$ is contained in R. Thus, the bipartite auxiliary graph B can be constructed in $O(s \cdot (|W| + |F|))$ time.

Consider lines 11 to 13. By choosing a "component vertex" x_{K_r} of minimum degree, we ensure that the set CC is a minimum-cardinality set of vertices from $N_H(v)$ separating at least one connected component in K from v. That is, CC separates the vertices in K_r from v. Let w be an arbitrary vertex of CC. By the deletion of all but one vertex from CC (line 13), we ensure that the graph $\hat{H} = H - (CC \setminus \{w\})$ is still connected and contains at least one cut-vertex, namely w. Hence, according to the observation above, $\hat{H}$ contains a vertex of degree at most s. Let v' be a minimum-degree vertex of $\hat{H}$ (line 14). As a consequence, $\deg_{\hat{H}}(v') \leq s$ and we can clearly find a set $S \subseteq V(\hat{H})$ of s vertices such that $H' := \hat{H}[N_{\hat{H}}[v'] \cup S]$ is connected. Note that H' is contained in $\mathcal{C}(s, \deg_{\hat{H}}(v')) \subseteq \mathcal{F}(s, s)$. Altogether, the running time is $O(s \cdot (|W| + |F|))$. □

Summarizing, we obtain a linear-time algorithm for finding a minimal forbidden

induced subgraph if s is a constant. In particular, this means we can find a forbidden induced subgraph comprising at most $s + \lfloor\sqrt{s}\rfloor$ vertices in linear time.

THEOREM 4.7. *Let $G = (V, E)$ be a graph that is not an s-plex cluster graph. Then, a minimal forbidden induced subgraph can be found in $O(s \cdot (n + m))$ time.*

Proof. Let $C = (W, F)$ be a connected component of G that is not an s-plex. Let v be a vertex of minimum degree in C. Clearly, by breadth-first search starting at v we can find a set $S \subseteq W$ of s vertices such that $H := G[N_G[v] \cup S]$ is connected. Note that $H \in \mathcal{C}(s, \deg_H(v))$. If $\deg_H(v) > s$, then we can apply Algorithm B (Figure 4.3) once to find a forbidden induced subgraph H' from $\mathcal{F}(s, s)$. In order to find a minimal forbidden induced subgraph, we apply Algorithm A (Figure 4.2) at most $O(s)$ times. Hence, the overall running time is $O(s \cdot (n + m))$. □

5. An Exact Search Tree Algorithm. In this section, we present a simple search tree algorithm that is based on the forbidden subgraph characterization from Section 4. Following Theorem 4.3, all minimal forbidden induced subgraphs for s-plex cluster graphs are contained in $\mathcal{F}(s, \lfloor -0.5 + \sqrt{0.25 + s}\rfloor)$. To obtain an s-plex cluster graph, every forbidden induced subgraph has to be destroyed via edge modifications. To this end, we apply a branching strategy.

THEOREM 5.1. *There is an $O((2s + \lfloor\sqrt{s}\rfloor)^k \cdot s \cdot (n + m))$–time algorithm solving s-PLEX CLUSTER EDITING.*

Proof. Given an instance (G, k) of s-PLEX CLUSTER EDITING, we search in G for a minimal forbidden induced subgraph from $\mathcal{F}(s, i)$ with $i = \lfloor -0.5 + \sqrt{0.25 + s}\rfloor$. By Theorem 4.7, this can be done in $O(s \cdot (n+m))$ time. If G does not contain an induced subgraph from $\mathcal{F}(s, i)$, then G already is an s-plex cluster graph and we are done. Otherwise, let S be a set of vertices inducing a forbidden subgraph $G[S] \in \mathcal{C}(s, i') \subseteq \mathcal{F}(s, i)$, where $i' \leq i$. In the following, let v denote a vertex with $\deg_{G[S]}(v) = i'$. By the definition of $\mathcal{C}(s, i')$, such a vertex must exist. Branch into the different possibilities to destroy the forbidden induced subgraph $G[S]$ and then recursively solve the instances that are created in the respective search tree branches. Clearly, the branching stops when $k \leq 0$.

For branching, either insert edges incident to v or delete edges in $G[S]$. It is sufficient to only consider these edge modifications since, if none of these is performed, then $G[S]$ remains connected and there are s vertices in $G[S]$ that are not adjacent to v, contradicting the s-plex (cluster graph) definition.

First, consider edge insertions between v and vertices $u \in S \setminus N[v]$. Since $G[S] \in \mathcal{C}(s, i')$ and $\deg_{G[S]}(v) = i'$, we have $|S \setminus N[v]| = s$. Therefore, branch into s cases, inserting a different edge in each search tree branch. The parameter decreases by 1 in each branch.

Next, consider edge deletions. In each remaining branch, there is at least one vertex $u \in S$ such that u and v become disconnected, that is, they are in different connected components of the final s-plex cluster graph. We now show that for each $u \in S$ it is sufficient to create one search tree branch in which at least one edge deletion is performed for the case that u and v are not connected in the final cluster graph. Let $S_l \subset S$ denote the vertices that have distance exactly l to v in $G[S]$. We first consider the vertices in S_1 (the neighbors of v in $G[S]$), then the vertices in S_2, and so on.

For each $u \in S_1$, create a search tree branch in which one disconnects u and v. Clearly this means that one has to delete the edge $\{u, v\}$. To branch on the vertices in S_2, one can assume that the vertices from $N[v] = \{v\} \cup S_1$ end up in the same cluster, since we have already considered all possibilities of removing edges between v

and the vertices in S_1. Therefore, when considering the case that a vertex $u \in S_2$ and v are not connected in the final cluster graph, one must delete all edges between u and its neighbors in S_1. At least one such edge must exist because $u \in S_2$. Therefore, for each case, one creates a search tree branch in which the parameter is decreased by at least 1.

The case distinction is performed for increasing values of l, always assuming that v and the vertices in $S_1 \cup S_2 \cup \ldots \cup S_{l-1}$ end up in the same cluster of the final cluster graph. Hence, when considering the case that v and a vertex $u \in S_l$ end up in different clusters, one creates a search tree branch in which the edges between u and its neighbors in S_{l-1} are deleted, and at least one of these edges must exist. Hence, one creates $|S| - 1 = s + i' \leq s + i$ branches in which edges are deleted. Together with the s cases in which edge insertions are performed, one branches into $2s + i$ cases, and in each branch, the parameter is decreased by at least 1. Branching is performed only as long as $k > 0$. Hence, the search tree has size $O((2s+i)^k) = O((2s + \lfloor\sqrt{s}\rfloor)^k)$, since $i = \lfloor -0.5 + \sqrt{0.25 + s}\rfloor$. Using breadth-first search, the steps at each search tree node can be performed in $O(s \cdot (n + m))$ time which results in the claimed running time bound. □

Using Theorems 3.4 and 5.1, by interleaving the problem kernelization (running in $O(n^4)$ time) and the search tree [27, 26], we get:

Theorem 5.2. *There is an $O((2s + \lfloor\sqrt{s}\rfloor)^k + n^4)$–time algorithm solving s-*Plex Cluster Editing.

6. Conclusion. We initiated the study of the graph modification problem s-Plex Cluster Editing. Whereas here we considered the edge modification scenario, follow-up work studied the vertex deletion variant [3]. We believe that s-Plex Cluster Editing may have practical relevance for graph-based data clustering in a similar way as its well-studied special case Cluster Editing has. Recently, there have been several further studies dealing with different relaxed models of graph-based data clustering [14, 19, 20]. Our results lead to numerous opportunities for future research. From the viewpoint of algorithm theory, we concentrated on fixed-parameter algorithms, leaving open the study of approximation algorithms. Second, we left unstudied the sometimes desirable case of having a specified number of clusters to be generated. As to applications, important issues of interest for future study would be to deal with weighted inputs or to obtain faster algorithms for special cases such as $s = 2$. A thorough empirical study as recently undertaken for Cluster Editing [6, 5, 11, 29, 34] is a natural task for future work. Finally, studying further parameterizations for s-Plex Cluster Editing as recently undertaken for Cluster Editing [21] seems promising.

Acknowledgement. We are grateful to Falk Hüffner for inspiring discussions in the early phase of this research and anonymous referees of *SIAM Journal on Discrete Mathematics* for detailed feedback improving our presentation.

References.

[1] B. Balasundaram, S. Butenko, and I. V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 2009. To appear.

[2] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1–3):89–113, 2004.

[3] R. van Bevern, H. Moser, and R. Niedermeier. Kernelization through tidying—a case study based on s-plex cluster vertex deletion. In *Proc. 9th LATIN*, volume 6034 of *LNCS*, pages 528–539. Springer, 2010.

[4] H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proc. 4th IWPEC*, volume 5917 of *LNCS*, pages 17–37. Springer, 2009.

[5] S. Böcker, S. Briesemeister, Q. B. A. Bui, and A. Truß. Going weighted: Parameterized algorithms for cluster editing. *Theoretical Computer Science*, 410(52): 5467–5480, 2009.

[6] S. Böcker, S. Briesemeister, and G. W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 2009. Available electronically.

[7] J. Chen and J. Meng. A $2k$ kernel for the cluster editing problem. In *Proc. 16th COCOON*, volume 6196 of *LNCS*, pages 459–468. Springer, 2010.

[8] 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, February 2005.

[9] S. Cohen, B. Kimelfeld, and Y. Sagiv. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *Journal of Computer and System Sciences*, 74(7):1147–1159, 2008.

[10] 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.

[11] F. Dehne, M. A. Langston, X. Luo, S. Pitre, P. Shaw, and Y. Zhang. The cluster editing problem: Implementations and experiments. In *Proc. 2nd IWPEC*, volume 4169 of *LNCS*, pages 13–24. Springer, 2006.

[12] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[13] M. R. Fellows, M. A. Langston, F. A. Rosamond, and P. Shaw. Efficient parameterized preprocessing for cluster editing. In *Proc. 16th FCT*, volume 4639 of *LNCS*, pages 312–321. Springer, 2007.

[14] M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Graph-based data clustering with overlaps. In *Proc. 15th COCOON*, volume 5609 of *LNCS*, pages 516–526. Springer, 2009. Long version submitted to *Discrete Optimization*.

[15] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[16] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, 38(4):373–392, 2005.

[17] J. Guo. A more effective linear kernelization for Cluster Editing. *Theoretical Computer Science*, 410(8-10):718–726, 2009.

[18] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.

[19] J. Guo, I. A. Kanj, C. Komusiewicz, and J. Uhlmann. Editing graphs into disjoint unions of dense clusters. In *Proc. 20th ISAAC*, volume 5878 of *LNCS*, pages 583–593. Springer, 2009.

[20] P. Heggernes, D. Lokshtanov, J. Nederlof, C. Paul, and J. A. Telle. Generalized graph clustering: recognizing (p, q)-cluster graphs. In *Proc. 36th WG*, LNCS. Springer, 2010. To appear.

[21] C. Komusiewicz and J. Uhlmann. Alternative parameterizations for cluster editing. Unpublished manuscript, September 2010.

[22] C. Komusiewicz, F. Hüffner, H. Moser, and R. Niedermeier. Isolation concepts

for efficiently enumerating dense subgraphs. *Theoretical Computer Science*, 410 (38–40):3640–3654, 2009.

[23] M. Křivánek and J. Morávek. NP-hard problems in hierarchical tree-clustering. *Acta Informatica*, 23(3):311–323, 1986.

[24] 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, 2007.

[25] H. Moser, R. Niedermeier, and M. Sorge. Algorithms and experiments for clique relaxations—finding maximum s-plexes. In *Proc. 8th SEA*, volume 5526 of *LNCS*, pages 233–244. Springer, 2009.

[26] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Number 31 in Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[27] R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73:125–129, 2000.

[28] F. Protti, M. D. da Silva, and J. L. Szwarcfiter. Applying modular decomposition to parameterized cluster editing problems. *Theory of Computing Systems*, 44(1): 91–104, 2009.

[29] S. Rahmann, T. Wittkop, J. Baumbach, M. Martin, A. Truß, and S. Böcker. Exact and heuristic algorithms for weighted cluster editing. In *Proc. 6th CSB*, volume 6 of *Computational Systems Bioinformatics*, pages 391–401. Imperial College Press, 2007.

[30] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[31] S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978.

[32] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1–2):173–182, 2004.

[33] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[34] T. Wittkop, J. Baumbach, F. P. Lobo, and S. Rahmann. Large scale clustering of protein sequences with FORCE – a layout based heuristic for weighted cluster editing. *BMC Bioinformatics*, 8(1):396, 2007.

[35] B. Wu and X. Pei. A parallel algorithm for enumerating all the maximal k-plexes. In *Proc. 11th PAKDD*, volume 4819 of *LNCS*, pages 476–483. Springer, 2007.

[36] R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[37] A. van Zuylen and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. *Mathematics of Operations Research*, 34(3):594–620, 2009.