A More Relaxed Model for Graph-Based Data Clustering: s-Plex Editing

Jiong Guo*, Christian Komusiewicz**, Rolf Niedermeier, and Johannes Uhlmann***

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany.
{jiong.guo,c.komus,rolf.niedermeier,johannes.uhlmann}@uni-jena.de

Abstract. We introduce the s-Plex Editing problem generalizing 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 (Cluster Editing), the task in the case of s-Plex Editing is now to transform a graph into a disjoint union of so-called s-plexes. Herein, an s-plex denotes a vertex set inducing a (sub)graph where every vertex has edges to all but at most s vertices in the s-plex. 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), which better reflects inaccuracies of the input data. We develop a provably efficient and 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 modification sets. Altogether, this yields efficient algorithms in case of moderate numbers of edge modifications.

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" [16]. To solve data clustering, one prominent line of attack is to use graph theory based methods [14]. In this line, extending and complementing previous work on cluster graph modification problems, we introduce the new edge modification problem s-Plex Editing.

In the context of graph-based clustering, data items are represented as vertices and there is an edge between two vertices iff the interrelation between the

* Partially supported by the DFG, Emmy Noether research group PIAF, NI 369/4, and research project DARE, GU 1023/1.
** Supported by a PhD fellowship of the Carl-Zeiss-Stiftung.
*** Supported by the DFG, research project PABI, NI 369/7.

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 no 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 [14]. Herein, a *cluster graph* is 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) [14, 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 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 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 [5] 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 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 editing operations. Note that 1-Plex Editing is the same as Cluster Editing. Compared to Cluster Editing, s-Plex 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. This important advantage of s-Plex 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. [3] for making this point in terms of Cluster Editing). Figure 1 presents a simple example comparing Cluster Editing (that is, 1-Plex Editing) with 2-Plex Editing and 3-Plex Editing in terms of the (number of) necessary editing operations.

Previous work and motivation. The s-plex concept was introduced in 1978 by Seidman and Foster [13] in the context of social network analysis. Recently, a

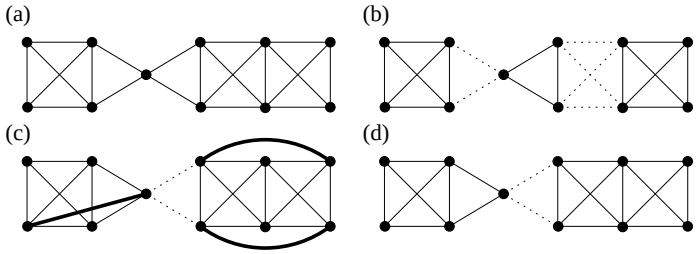

Fig. 1. An example for different optimal modifications that are applied to (a) an input graph using (b) Cluster Editing (equivalently, 1-Plex Editing), (c) 2-Plex Editing, and (d) 3-Plex Editing. Deleted edges are dashed, inserted edges are bold.

number of theoretical and experimental studies explored (and confirmed) the usefulness of s-plexes in various contexts [1, 6, 10, 11]. Finding maximum-cardinality s-plexes is NP-hard [1] and further hardness results in analogy to clique finding hold as well [10]. Hence, there is no hope for polynomial-time algorithms.

Cluster Editing has recently been intensively studied from the viewpoints of polynomial-time approximability as well as parameterized algorithmics. As to approximability, the currently best known approximation factor is 2.5 [17]. Considering the parameter k defined as the number of allowed edge modifications, a search tree of size $O(1.83^k)$ [3] has been developed and several studies concerning provably efficient and effective preprocessing by data reduction (which is called problem kernelization in the context of parameterized algorithmics [12]) have been performed [7, 8]. Parameterized algorithms have led to several successful experimental studies mainly in the context of biological network analysis [3, 4]. 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. We mention in passing that slightly modifying a proof of Shamir et al. [14] for Cluster Editing one can show that s-Plex Editing is NP-complete for each specific choice of s as well.

Our contributions. We develop a polynomial-time preprocessing algorithm that allows to provably simplify input instances of s-Plex Editing to smaller ones. More specifically, the corresponding data reduction rules, given an instance $(G = (V, E), k)$ of s-Plex Editing with $s \geq 2$, in polynomial time construct an equivalent reduced instance $(G' = (V', E'), k')$ with $V' \subseteq V$, $k' \leq k$, and $|V'| \leq (4s^2-2)\cdot k+4(s-1)^2$. 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 loosing information. In terms of parameterized algorithmics, the reduced instance gives a problem kernel. Moreover, we provide a graph-theoretic characterization of s-plex cluster graphs by means of forbidden

induced subgraphs. In particular, we obtain a linear-time recognition algorithm for s-plex cluster graphs for every constant s. This is a result of independent graph-theoretic interest and is also of decisive algorithmic use for clustering: Based on the forbidden subgraph characterization of s-plex cluster graphs, we show that s-Plex 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). Moreover, interleaving the problem kernelization and the search tree leads to a running time of $O((2s + \lfloor\sqrt{s}\rfloor)^k + |V|^4)$.

Due to the lack of space, many technical details are deferred to a full version of this paper.

2 Preliminaries

We only consider *undirected* graphs $G = (V, E)$, where $n := |V|$ and $m := |E|$. 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.

Parameterized algorithmics [12] 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* [9, 12]. 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 iff (G', k') is a yes-instance.

3 Data Reduction and Kernelization

In this section, we indicate that s-Plex Editing for $s \geq 2$ admits a problem kernel with $(4s^2 - 2) \cdot k + 4(s-1)^2$ vertices. Since s-Plex Editing is a generalization of Cluster Editing, the first idea coming to mind in order to achieve

a linear kernelization is to adapt an approach developed by Guo [8]. However, since the "critical clique" concept used there does not work for s-Plex Editing, we need a more sophisticated strategy; correspondingly, 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 connected components that are s-plexes from G.

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 weakly connected to the rest of the graph. Unfortunately, the proof details are technical and require quite some mathematical machinery.

We start with explaining in more detail the purpose of introducing the second data reduction rule. Let G_{opt} denote the s-plex cluster graph resulting from applying a solution S with $|S| \leq k$ to the graph $G = (V, E)$, and let $K_1, \ldots, K_l$ be the s-plexes in G_{opt}. 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, $|X| \leq 2k$. To achieve a problem kernel with $O(k)$ vertices for constant s, it remains to bound $|Y|$. To this end, we use a function linear in $|X_i|$ to bound $|Y_i|$ for each $1 \leq i \leq l$. If $|Y_i| \leq (s-1) \cdot |X_i|$ or $|Y_i| \leq 2(s-1)$ for all i, then we are done; otherwise, we have at least one 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 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 every vertex in Y_i has, in G, distance at most two to u: Suppose that this is not true. Let x be a vertex in Y_i with distance at least three to u. Then, since K_i is an s-plex, we must have $|Y_i \setminus N_G[u]| \leq s-1$ (since otherwise u would be non-adjacent to more than $s-1$ vertices) as well as $|N_G[u] \cap Y_i| \leq s-1$ (since otherwise x would be non-adjacent to more than $s-1$ vertices—the vertices $N_G[u] \cap Y_i$ are non-adjacent to x since x has distance at least three to u), contradicting $|Y_i| > 2(s-1)$.

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_{opt}, 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,

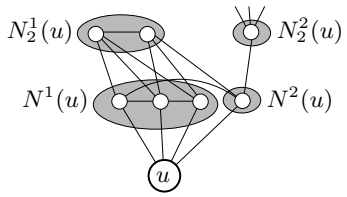

Fig. 2. An illustration of the partitions of $N_2(u)$ and $N(u)$ for $\hat{s} = 2$. Vertex u satisfies the first two preconditions of Reduction Rule 2.

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:

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

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 2 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 which shrinks the set of the vertices tightly coupled with a vertex u that has a very special neighborhood: There can be many vertices in $N^1(u)$, but only few (at most $\hat{s}$) tightly coupled vertices from $N_2(u)$, that is, the $N_2^1(u)$-vertices. Further, these $N_2^1(u)$-vertices are only adjacent to vertices in $N(u)$ or to $N_2^1(u)$-vertices. Reduction Rule 2 applies in this situation and replaces $N^1(u) \cup N_2^1(u)$ by smaller "simulating" cliques.

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) $|A| > \alpha$, where $A := \{u\} \cup N^1(u) \cup N_2^1(u)$ and $\alpha := 2\,\hat{s} \cdot (|N^2(u)| + |N_2^2(u)| + \hat{s})$,

then replace A by a clique C with α vertices for even $|A|$ or $\alpha + 1$ vertices for odd $|A|$. Further, 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 $\max\{|A_v| - |C_v|, 0\}$.

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 iff the graph G' resulting by 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. The first one (Lemma 1) says that if a vertex u satisfies the three preconditions of Reduction Rule 2 then all vertices in A as defined in the rule should be completely contained in one s-plex K in the s-plex cluster graph generated by some optimal solution and $K \subseteq A \cup N^2(u)$. The second claim (Lemma 2) says that, after replacing A by a clique C, all vertices in C are tightly coupled with the vertices in $N^2(u)$. Using the second claim, we can show that there must be a vertex in C satisfying the preconditions of the first claim. Thus, according to the first claim, there exists an optimal solution of G' which generates an s-plex K' with $C \subseteq K'$ and $K' \subseteq C \cup N^2(u)$. Then, by focussing on the edge modifications in S and S' that generate K and K', respectively, we can directly compare the sizes of S and S' and, thereby, establish the "iff"-relation between G and G'.

Lemma 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 always an optimal solution generating an s-plex cluster graph where*
(1) the set A is completely contained in one s-plex K and
(2) $K \subseteq A \cup N^2(u)$.

Now, before coming to the second claim (Lemma 2), we discuss in more detail the strategy behind. Based on Lemma 1, we can conclude that, with respect to a vertex u which satisfies the preconditions of Reduction Rule 2, it remains to decide which vertices of $N^2(u)$ should build, 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 in 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 should 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. The second claim shows then that C plays the same role as A.

Lemma 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 has at most $\hat{s}$ non-adjacent vertices in $N_{G'}(C)$.*

With these two claims, we can prove the correctness and the running time of Reduction Rule 2.

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

Finally, we prove the main theorem in this section. Note that the running time upper bound is a pure worst-case estimation; improvements are conceivable.

Theorem 1. s-Plex Editing *admits a problem kernel with $(4s^2-2)\cdot k+4(s-1)^2$ vertices for $s \geq 2$. It can be computed in $O(n^4)$ time.*

Proof. Let G_{opt} 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_{opt}. The vertices V of G_{opt} 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 conditions C1–C4. By $N_2^1(u) \subseteq Y_i$ and $N[u] \subseteq K_i$, vertex u fulfils the first two preconditions of Reduction Rule 2. Since $|Y_i| \leq |N^1(u) \cup N_2^1(u) \cup \{u\}|$, this implies either $|Y_i| \leq \alpha := 2\,\hat{s}\cdot(|N^2(u)| + |N_2^2(u)| + \hat{s})$ 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. Note that $N^2(u) \cup N_2^2(u) \subseteq X$ and, for every deleted 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 s-plexes, we then have

$$\sum_{1\leq i\leq l} |Y_i| \leq \sum_{1\leq i\leq l} \max\{2\,\hat{s}, \hat{s}\cdot|X_i|, 4\,\hat{s}\cdot(|X_i| + \hat{s})\} \overset{(***)}{\leq} 8\,\hat{s}\,k + 4\,\hat{s}^2\cdot(k+1).$$

The 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 $|X| + |Y| \leq 4\,\hat{s}^2(k+1) + 8\,\hat{s}\,k + 2k = (4s^2-2)k + 4(s-1)^2$ vertices.

The running time $O(n^4)$ follows directly from Lemma 3 and the fact that Reduction Rule 2 can be applied at most n times. □

4 Forbidden Subgraph Characterization and Search Tree

This section presents a forbidden subgraph characterization of s-plex cluster graphs for any $s \geq 1$ as well as an exact search tree algorithm that makes use

Input: $G = (V, E)$ from $\mathcal{C}(s, i)$ with $i \cdot (i + 1) > s$
Output: An induced subgraph $G' \in \mathcal{C}(s, i')$ of G with $i' < i$

Let $v = \text{argmin}_{w \in V}\{\deg_G(w)\}$
if $\deg_G(v) < i$ **then**
    return a connected graph induced by $N_G[v]$ and further s arbitrary vertices
Let Cutvertices be the set of cut-vertices of G
if $(N_G(v) \setminus \text{Cutvertices}) \neq \emptyset$ **then**
    return graph $G - w$ for an arbitrary $w \in (N_G(v) \setminus \text{Cutvertices})$
Let $N_G(v) = \{u_1, u_2, \ldots, u_i\}$
Let $U_j \subseteq V$ be the vertices not reachable from v in $G - u_j$, for $1 \leq j \leq i$
Let $r = \text{argmin}_{j=1,\ldots,i}\{|U_j|\}$
return $G - (U_r \setminus \{w\})$ for an arbitrary vertex $w \in U_r$

Fig. 3. Algorithm A to compute smaller forbidden subgraphs.

of this characterization. We provide a characterization of s-plex cluster graphs by means of *induced* forbidden subgraphs. More specifically, we specify a set $\mathcal{F}$ of graphs such that a graph G is an s-plex cluster graph iff 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 subgraph is a path induced by three vertices [14]. By way of contrast, if $s \geq 2$, we face up to exponentially in s many forbidden subgraphs. To cope with this, we develop a characterization of these subgraphs that still allows us to derive efficient algorithms. More specifically, we show that s-plex cluster graphs are characterized by forbidden subgraphs with $O(s)$ vertices and that if a graph is not an s-plex cluster graph then a forbidden subgraph can be found in $O(s \cdot (n+m))$ time.

The starting point for the forbidden subgraph characterization are the connected graphs that contain a vertex that is non-adjacent to s vertices. These graphs clearly are no s-plex cluster graphs. Let $\mathcal{C}$ denote the set of connected graphs. Define $\mathcal{C}(s, i) := \{G = (V, E) \in \mathcal{C} \mid (|V| = s+1+i) \wedge (\exists v \in V : \deg_G(v) = i)\}$ and $\mathcal{F}(s, i) := \bigcup_{j=1}^{i} \mathcal{C}(s, j)$. The following lemma shows that the graphs in $\mathcal{F}(s, n - s - 1)$ are forbidden.

Lemma 4. *A graph G is an s-plex cluster graph iff G is $\mathcal{F}(s, n - s - 1)$-free.*

Next, we show that instead of studying graphs with $O(n)$ vertices, we can focus on graphs with $O(s)$ vertices by presenting an algorithm (Algorithm A, see Fig. 3) shrinking the size of large forbidden subgraphs. More precisely, we show that if the forbidden subgraph $G \in \mathcal{C}(s, i)$ with $i \cdot (i + 1) > s$ then we can always remove at least one vertex from G and still obtain a forbidden induced subgraph. For brevity, let T_s be the maximum integer satisfying $T_s \cdot (T_s + 1) \leq s$, that is, $T_s = \lfloor -0.5 + \sqrt{0.25 + s} \rfloor$.

Lemma 5. *Given a graph $G = (V, E) \in \mathcal{C}(s, i)$ such that $i > T_s$, Algorithm A (Fig. 3) computes in $O(|V| + |E|)$ time an induced subgraph $G' \in \mathcal{C}(s, i')$ of G, with $i' < i$.*

Proof. Consider lines 1 to 3 of the algorithm. If a vertex v in G has degree less than i, then we can clearly find a graph from $\mathcal{C}(s, \deg_G(v))$ by choosing $N_G[v]$ and a set $S \subseteq V$ of s further (arbitrary) vertices such that $G[N_G[v] \cup S]$ 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 G obtaining a graph from $\mathcal{C}(s, i-1)$. Note that cut-vertices can be computed in linear time [15].

Consider lines 7 to 9. All neighboring vertices $N_G(v) = \{u_1, u_2, \ldots, u_i\}$ of v are cut-vertices and the minimum vertex degree is i with $i \cdot (i+1) > s$. On the one hand, note that $|U_j| \geq i$ for every $1 \leq j \leq i$ since the minimum vertex degree of G is i and since for every vertex $w \in U_j$ it holds that $N_G(w) \subseteq (U_j \cup \{u_j\}) \setminus \{w\}$. On the other hand, since $\sum_{j=1}^{i} |U_j| \leq s$, 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 G 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 from $\mathcal{C}(s, 1)$. Note that Tarjan's algorithm [15] also computes a so-called *block-tree*. With the help of this data structure, the sets U_j can be easily computed in linear time. □

We can iteratively use Algorithm A to compute an induced subgraph of G' from $\mathcal{C}(s, i')$ with $i' \leq T_s$. This results in the following forbidden subgraph characterization that is—in contrast to the one of Lemma 4—tight concerning the number of vertices of the forbidden subgraphs.

Theorem 2. *A graph $G = (V, E)$ is an s-plex cluster graph if and only if G is $\mathcal{F}(s, T_s)$-free.*

Proof. On the one hand, due to Lemma 4 we know that an s-plex cluster graph is $\mathcal{F}(s, n-s-1)$-free and, hence, $\mathcal{F}(s, T_s)$-free. On the other hand, if G contains a forbidden subgraph from $\mathcal{C}(s, i')$ with $i' \cdot (i'+1) > s$, and, hence, according to Lemma 4 is not an s-plex cluster graph, then we can iteratively use Algorithm A (Figure 3) to find a forbidden subgraph from $\mathcal{C}(s, i)$ with $i \leq T_s$. □

To show our main result, we develop an $O(s \cdot (n+m))$-time algorithm (Algorithm B, see Figure 4) to find a forbidden 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 forbidden subgraph from $\mathcal{F}(s, T_s)$. Overall, this approach yields linear-time for any constant s.

Lemma 6. *Algorithm B (Figure 4) is correct and has running time $O(s \cdot (|V| + |E|))$.*

Proof. Consider lines 1 to 3. If $\deg_G(u) \leq s$, then we can clearly find a set $S \subseteq V$ of s vertices such that $G[N_G[u] \cup S]$ is connected. This graph is in $\mathcal{C}(s, i')$ for an $i' \leq s$.

In the following, we need the observation that if one of the neighboring vertices of v is a cut-vertex, then there exists at least one vertex in G' with degree at most s. This can be seen as follows. Assume that $x \in N_G(v)$ is a cut-vertex

Input: $G = (V, E)$ from $\mathcal{C}(s, i)$ with $i > s$
Output: An induced subgraph $G' \in \mathcal{C}(s, i')$ of G with $i' \leq s$

1 Let $u = \text{argmin}_{w \in V'}\{\deg_G(w)\}$
2 **if** $\deg_G(u) \leq s$ **then**
3 **return** a connected graph induced by $N_G[u]$ and further s arbitrary vertices
4 Let $v \in V'$ be a vertex with $\deg_G(v) = i$
5 Let $N_G(v) = \{u_1, u_2, \ldots, u_i\}$
6 Let $K = \{K_1, K_2, \ldots, K_l\}$ with $l \leq s$
  denote the connected components of $G - N_G[v]$
7 Construct an auxiliary bipartite graph $H = (W_N, W_K, F)$ with
8 $W_N := \{w_{u_j} \mid 1 \leq j \leq i\}$,
9 $W_K := \{w_{K_q} \mid 1 \leq q \leq l\}$, and
10 $F := \{\{w_{u_j}, w_{K_q}\} \mid \exists\{u_j, v'\} \in E \text{ with } v' \in K_q\}$
11 Let $r := \arg\min_{q=1,\ldots,l}\{\deg_H(w_{K_q})\}$
12 Let $CC := \{u_j \mid w_{u_j} \in N_H(w_{K_r})\}$
13 Let $\hat{G} = G - (CC \setminus \{w\})$ for an arbitrary vertex $w \in CC$
14 Let $v' = \text{argmin}_{w \in V(\hat{G})}\{\deg_{\hat{G}}(w)\}$
15 **return** a connected graph induced by $N_{\hat{G}}[v']$ and further s arbitrary vertices

Fig. 4. Algorithm B to compute in linear time a forbidden subgraph with $O(s)$ vertices.

and let $U \subseteq V$ denote the vertices not reachable from v in $G - x$. Since a vertex $w \in U$ can only be adjacent to vertices in $U \cup \{x\}$ and $|U| \leq s$, we have that $\deg_G(w) \leq s$.

According to this observation, when entering line 5 of Algorithm B, we know that none of the vertices in $N_G(v) = \{u_1, u_2, \ldots, u_i\}$ 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 $N_G(v)$ whose removal leads to a connected graph in which one neighbor of v is a cut-vertex. To this end, one builds an auxiliary bipartite graph $H = (W_N, W_K, F)$ (lines 5-10). As to the running time needed for the construction of H, note that the degree of a vertex in W_N is at most s since $G - (N_G(v) \cup \{v\})$ contains exactly s vertices and, hence, W_K has size at most s. Thus, to construct F, we can iterate over the edge set E and, given an edge $\{u_j, v'\}$ with $v' \in K_q$, we can decide in $O(s)$ time whether the edge $\{w_{u_j}, w_{K_q}\}$ is contained in F. Thus, the bipartite auxiliary graph H can be constructed in $O(s \cdot (|V| + |E|))$ time.

Consider lines 11 to 13. By choosing a "component vertex" w_{K_r} of minimum degree, we ensure that the set CC is a minimum-cardinality set of vertices from $N_G(v)$ separating at least one connected component in K from v. In particular, 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 $\hat{G} = G - (CC \setminus \{w\})$ is still connected and contains at least one cut-vertex, namely w. Hence, according to the observation above, $\hat{G}$ contains a vertex of degree at most s. Let v' be a minimum-degree vertex of $\hat{G}$ (line 14).

As a consequence, $\deg_{\hat{G}}(v') \leq s$ and we can clearly find a set $S \subseteq V(\hat{G})$ of s vertices such that $G' := \hat{G}[N_{\hat{G}}[v'] \cup S]$ is connected. Note that G'' is contained in $\mathcal{C}(s, \deg_{\hat{G}}(v')) \subseteq \mathcal{F}(s, s)$.

Altogether, the running time is $O(s \cdot (|V| + |E|))$. □

Summarizing, we obtain a linear-time algorithm for finding an induced forbidden subgraph if s is a constant.

Theorem 3. *Let $G = (V, E)$ be a graph that is not an s-plex cluster graph. Then, a forbidden subgraph from $\mathcal{F}(s, T_s)$ 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 $G' := G[N_G[v] \cup S]$ is connected. Note that $G' \in \mathcal{C}(s, \deg_{G'}(v))$. If $\deg_{G'}(v) > s$, then we can apply Algorithm B (Figure 4) once to find an induced forbidden subgraph G'' from $\mathcal{F}(s, s)$. In order to find a forbidden subgraph from $\mathcal{F}(s, T_s)$, we apply Algorithm A (Figure 3) at most $O(s)$ times. □

Next, we present a search tree algorithm that is based on this forbidden subgraph characterization. To obtain an s-plex cluster graph, every forbidden subgraph has to be destroyed via edge modifications. To this end, we apply a branching strategy.

Theorem 4. *s-Plex Editing can be solved in $O((2s + \lfloor\sqrt{s}\rfloor)^k \cdot s \cdot (n + m))$ time.*

Proof. Given an instance (G, k) of s-Plex Editing, we search in G for a forbidden subgraph from $\mathcal{F}(s, T_s)$. By Theorem 3, this can be done in $O(s \cdot (n+m))$ time. If G does not contain a 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 $i' \leq T_s$. 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. We now branch into the different possibilities to destroy the forbidden subgraph $G[S]$ and then recursively solve the instances that are created in the respective search tree branches.

For branching, we 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, we 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, we branch into s cases, inserting a different edge in each search tree branch. The parameter decreases by 1 in each branch.

Besides this, we consider edge deletions. Hence, in each remaining branch, there is at least one vertex $u \in S$ such that u and v are not connected, that is, they are in different connected components of the final s-plex cluster graph. We

now show that for each $u \in S$ we can create a 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 $u \in S_1$, we create a search tree branch in which we disconnect u and v. Clearly this means that we have to delete the edge $\{u, v\}$. To branch on the vertices in S_2, we 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, we 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, we create 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, we create 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, we create $|S| - 1 = s + i' \leq s + T_s$ branches in which edges are deleted. Together with the s cases in which edge insertions are performed, we branch into $2s + T_s$ cases, and in each branch, the parameter is decreased by at least 1. Branching is performed only as long as $k > 0$. The search tree thus has size $O((2s + T_s)^k) = O((2s + \lfloor\sqrt{s}\rfloor)^k)$. 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 1 and 4, by interleaving the problem kernelization and the search tree [12], we get:

Theorem 5. *s-Plex Editing can be solved in $O((2s + \lfloor\sqrt{s}\rfloor)^k + n^4)$ time.*

5 Conclusion

We initiated the study of the graph modification problem s-Plex Editing. We believe that s-Plex Editing may have practical relevance for graph-based data clustering in a similar way as its well-studied special case Cluster Editing. Our results lead to numerous opportunities for future research. First, from the viewpoint of algorithm theory, we concentrated on parameterized 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 try to obtain faster algorithms for special cases such as $s = 2$. A thorough empirical study as recently undertaken for Cluster Editing [4] is a natural next step for future work.

Acknowledgement. We are grateful to Falk Hüffner for inspiring discussions in the early phase of this research.

References

[1] B. Balasundaram, S. Butenko, I. V. Hicks, and S. Sachdeva. Clique relaxations in social network analysis: The maximum k-plex problem, 2006. Manuscript.

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

[3] S. Böcker, S. Briesemeister, Q. B. A. Bui, and A. Truß. Going weighted: Parameterized algorithms for cluster editing. In *Proc. 2nd COCOA*, volume 5165 of *LNCS*, pages 1–12. Springer, 2008.

[4] S. Böcker, S. Briesemeister, and G. W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. In *Proc. 7th WEA*, volume 5038 of *LNCS*, pages 289–302. Springer, 2008.

[5] E. J. Chesler, L. Lu, S. Shou, Y. Qu, J. Gu, J. Wang, H. C. Hsu, J. D. Mountz, N. E. Baldwin, M. A. Langston, D. W. Threadgill, K. F. Manly, and R. W. Williams. Complex trait analysis of gene expression uncovers polygenic and pleiotropic networks that modulate nervous system function. *Nature Genetics*, 37(3):233–242, February 2005.

[6] V. J. Cook, S. J. Sun, J. Tapia, S. Q. Muth, D. F. Argüello, B. L. Lewis, R. B. Rothenberg, P. D. McElroy, and the Network Analysis Project Team. Transmission network analysis in tuberculosis contact investigations. *Journal of Infectious Diseases*, 196:1517–1527, 2007.

[7] 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.

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

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

[10] C. Komusiewicz, F. Hüffner, H. Moser, and R. Niedermeier. Isolation concepts for enumerating dense subgraphs. In *Proc. 13th COCOON*, volume 4598 of *LNCS*, pages 140–150. Springer, 2007.

[11] 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.

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

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

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

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

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

[17] A. van Zuylen and D. P. Williamson. Deterministic algorithms for rank aggregation and other ranking and clustering problems. In *Proc. 5th WAOA*, volume 4927 of *LNCS*, pages 260–273. Springer, 2008.