Lossy Kernels for Graph Contraction Problems

R. Krithika[1], Pranabendu Misra[2], Ashutosh Rai[3], and Prafullkumar Tale[4]

1 **The Institute of Mathematical Sciences, HBNI, Chennai, India**
`rkrithika@imsc.res.in`
2 **The Institute of Mathematical Sciences, HBNI, Chennai, India**
`pranabendu@imsc.res.in`
3 **The Institute of Mathematical Sciences, HBNI, Chennai, India**
`ashutosh@imsc.res.in`
4 **The Institute of Mathematical Sciences, HBNI, Chennai, India**
`pptale@imsc.res.in`

Abstract

We study some well-known graph contraction problems in the recently introduced framework of *lossy kernelization*. In classical kernelization, given an instance (I, k) of a parameterized problem, we are interested in obtaining (in polynomial time) an equivalent instance (I', k') of the same problem whose size is bounded by a function in k. This notion however has a major limitation. Given an approximate solution to the instance (I', k'), we can say nothing about the original instance (I, k). To handle this issue, among others, the framework of lossy kernelization was introduced. In this framework, for a constant α, given an instance (I, k) we obtain an instance (I', k') of the same problem such that, for every $c > 1$, any c-approximate solution to (I', k') can be turned into a $(c\alpha)$-approximate solution to the original instance (I, k) in polynomial time. Naturally, we are interested in a polynomial time algorithm for this task, and further require that $|I'| + k' = k^{\mathcal{O}(1)}$. Akin to the notion of polynomial time approximation schemes in approximation algorithms, a parameterized problem is said to admit a polynomial size approximate kernelization scheme (PSAKS) if it admits a polynomial size α-approximate kernel for every approximation parameter $\alpha > 1$. In this work, we design PSAKSs for Tree Contraction, Star Contraction, Out-Tree Contraction and Cactus Contraction problems. These problems do not admit polynomial kernels, and we show that each of them admit a PSAKS with running time $k^{f(\alpha)}|I|^{\mathcal{O}(1)}$ that returns an instance of size $k^{g(\alpha)}$ where $f(\alpha)$ and $g(\alpha)$ are constants depending on α.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases parameterized complexity, lossy kernelization, graph theory, edge contraction problems

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2016.23

1 Introduction

Many computational problems arising from real-world problems are NP-hard, and we do not expect any efficient algorithms for solving them optimally. Preprocessing heuristics, or data reduction rules, are widely applied to reduce large instances of these problems to a smaller size before attempting to solve them. Such algorithms are often extremely effective, and provide a significant boost to the subsequent step of computing a solution to the instance. Kernelization, under the aegis of Parameterized Complexity, has been developed as a mathematical framework to study these algorithms and quantify their efficacy.

In *Parameterized Complexity*, we consider instances (I, k) of a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. Typically, I is an instance of some computational problem, and k denotes the *parameter* which reflects some structural property of the instance. A common parameter is a bound on the size of an optimum solution to the problem instance. A data reduction algorithm, formally called a *Kernelization algorithm*, runs in polynomial time and reduces a given instance (I, k) of the problem to an *equivalent* instance (I', k') such that $|I'| + k' = k^{\mathcal{O}(1)}$. The instance (I', k') is called a *polynomial kernel*, and we say that the problem Π admits a polynomial kernelization (also called classical kernelization). Desigining kernelization algorithms for various computational problems, and investigating the associated lower bounds, is an active area of research in Computer Science. We refer the reader to [6, 9, 10] for an introduction to Parameterized Complexity and Kernelization.

The notion of polynomial kernels turns out to be a bit stringent, and it has been discovered that many problems do not admit a polynomial kernel under well-known complexity theoretic conjectures. On the other hand this notion turns out to be too lax as the instances (I, k) and (I', k') are not as tightly-coupled as one would like. For example, it is not possible to translate an approximate solution to the instance (I', k'), into an approximate solution to the original instance (I, k). Indeed, given anything but an optimal solution (or a solution of size k') to (I', k'), it is impossible to conclude anything about the original instance (I, k). These issues, among others, have led to the development of a framework for "approximation preserving kernelization" or *Lossy Kernelization.* Informally, an α-approximate kernelization algorithm ensures that given any c-approximate solution to the kernel (I', k'), it can be converted into a $c \cdot \alpha$-approximate solution to the original instance (I, k) in polynomial time. This notion was formally introduced, very recently, in [18] which shows that there are many problems without classical polynomial kernels that admit lossy polynomial kernels. Furthermore, it is likely that this notion will be very useful in practice. Many state of the art approximation algorithms are extremely sophisticated and it is infeasible to apply them to large problem instances. It is far more practical to reduce a large instance to a small kernel, then obtain a good approximate solution to this kernel, and finally transform it into an approximate solution to the original instance. In other words, lossy kernelization provides a mathematical framework for designing and analyzing preprocessing heuristics for approximation algorithms.

Let us state these notions formally. We first define a *parameterized optimization (maximization / minimization) problem*, which is the parameterized analogue of an optimization problem in the theory of approximation algorithms. A *parameterized minimization problem* is a computable function $\Pi : \Sigma^* \times \mathbb{N} \times \Sigma^* \mapsto \mathbb{R} \cup \{\pm\infty\}$. The instances of Π are pairs $(I, k) \in \Sigma^* \times \mathbb{N}$ and a solution to (I, k) is simply a string $S \in \Sigma^*$ such that $|S| \leq |I| + k$. The *value* of a solution S is $\Pi(I, k, S)$. The *optimum value* of (I, k) is $\mathrm{OPT}_\Pi(I, k) = \min_{S \in \Sigma^*, |S| \leq |I|+k} \Pi(I, k, S)$, and an *optimum solution* for (I, k) is a solution S such that $\Pi(I, k, S) = \mathrm{OPT}_\Pi(I, k)$. A *parameterized maximization problem* is defined in a similar way. We will omit the subscript Π in the notation for optimum value if the problem under consideration is clear from context. Next we come to the notion of an *α-approximate polynomial-time preprocessing algorithm* for a parameterized optimization problem Π. It is defined as a pair of polynomial-time algorithms, called the *reduction algorithm* and the *solution lifting algorithm*, that satisfy the following properties.

- Given an instance (I, k) of Π, the reduction algorithm computes an instance (I', k') of Π.
- Given the instances (I, k) and (I', k') of Π, and a solution S' to (I', k'), the solution lifting algorithm computes a solution S to (I, k) such that $\frac{\Pi(I,k,S)}{\mathrm{OPT}(I,k)} \leq \alpha \cdot \frac{\Pi(I',k',S')}{\mathrm{OPT}(I',k')}$.

A *reduction rule* is the execution of the reduction algorithm on an instance, and it is applicable on an instance if the output instance is different from the input instance. An

α-approximate kernelization (or α-approximate kernel) for Π is an α-approximate polynomial-time preprocessing algorithm such that the size of the output instance is upper bounded by a computable function $g : \mathbb{N} \times \mathbb{N}$ of k. In classical kernelization, often we apply reduction rules several times to reduce the given instance. This however breaks down in lossy kernelization, since each application of a reduction rule increases the "gap" between the approximation quality of the solution to the kernel, and the approximation quality of solution to the original instance that is computed by the solution lifting algorithm. To remedy this shortcoming, we require the notion of α-strict kernelization and α-safe reduction rules. An α-approximate kernelization is said to be *strict* if $\frac{\Pi(I,k,s)}{\mathrm{OPT}(I,k)} \leq \max\{\frac{\Pi(I',k',s')}{\mathrm{OPT}(I',k')}, \alpha\}$. A reduction rule is said to be *α-safe* for Π if there is a solution lifting algorithm, such that the rule together with this algorithm constitute a strict α-approximate polynomial-time preprocessing algorithm for Π. A reduction rule is *safe* if it is 1-safe, and note this this is more strict that the usual definition of safeness in classical kernelization. A *polynomial-size approximate kernelization scheme (PSAKS)* for Π is a family of α-approximate polynomial kernelization algorithms for each $\alpha > 1$. Note that, the size of an output instance of a PSAKS, when run on (I, k) with approximation parameter α, must be upper bounded by $f(\alpha)k^{g(\alpha)}$ for some functions f and g independent of $|I|$ and k. And finally, let us discuss the importance of the parameter k in this framework. In a classical kernelization, given an instance (I, k) the algorithm either returns another instance (i.e. a kernel) or decides that given instance has no solution of value at most k (i.e a NO instance). In lossy kernelization the output is always an instance of the optimization problem, and we must ensure that our kernelization algorithm must be safe on all instances. This may seem like a difficult goal, but recall that we are only interested in solutions of value at most k, and therefore we may define our parameterized minimization problem to reflect this fact.

$$\Pi(I,k,S) = \begin{cases} \infty & \text{if S is not a solution} \\ \min\{|S|, k+1\} & \text{otherwise} \end{cases}$$

This definition allows us to design the reduction rules without regard to the solutions of value more than k. In particular, if the solution lifting algorithm is given a solution of value $k + 1$ or more, it simply returns an a trivial feasible solution to the instance, and this is safe as per the above definitions. We encourage the reader to see [18] for a more comprehensive discussion of these ideas and definitions.

In [18], the authors exhibit lossy kernels for several problems which do not admit a classical kernelization, such as CONNECTED VERTEX COVER, DISJOINT CYCLE PACKING and DISJOINT FACTORS. They also develop a lower bound framework for lossy kernels, by extending the lower bound framework of classical kernelization. They then show that LONGEST PATH does not admit a lossy kernel of polynomial size unless NP $\subseteq$ coNP/poly. In this paper, we investigate several other problems in the framework of lossy kernelization. In particular, we design lossy polynomial kernels for several *graph contraction* problems which do not admit classical polynomial kernels under well known complexity theoretic conjectures. These problems are defined as follows. For a graph class $\mathcal{G}$, the $\mathcal{G}$-CONTRACTION problem is to determine if an input graph G can be contracted to some graph $H \in \mathcal{G}$ using at most k edge contractions. These problems are well studied and $\mathcal{G}$-CONTRACTION has been proven to be NP-complete for several classes $\mathcal{G}$ [1, 4, 20, 21]. They have also received a lot of attention in Parameterized Complexity [2, 5, 12, 13, 14, 15, 16, 17, 19]. In this work, we give lossy polynomial kernels for the following problems. In the following G/F denotes the graph obtained from G by contracting the edges in F.

Tree Contraction **Parameter:** k
Input: A graph G and an integer k
Question: Does there exist $F \subseteq E(G)$ of size at most k such that G/F is a tree?

Star Contraction **Parameter:** k
Input: A graph G and an integer k
Question: Does there exist $F \subseteq E(G)$ of size at most k such that G/F is a star?

Out-Tree Contraction **Parameter:** k
Input: A digraph D and an integer k
Question: Does there exist $F \subseteq A(D)$ of size at most k such that D/A is an out-tree?

Cactus Contraction **Parameter:** k
Input: A graph G and an integer k
Question: Does there exist $F \subseteq E(G)$ of size at most k such that G/F is a cactus?

It can be shown that these problems do not admit polynomial kernels, via a parameter preserving reduction from the Red Blue Dominating Set problem. Let us define these terms formally. A *polynomial-time parameter preserving reduction* from a parameterized problem Π_1 to a parameterized problem Π_2 is a polynomial-time function that maps an instance (I_1, k_1) of Π_1 to an instance (I_2, k_2) of Π_2 such that $k_2 = k_1^{\mathcal{O}(1)}$, and (I_1, k_1) is an YES instance of Π_1 if and only if (I_2, k_2) is an YES instance of Π_2. It is known that if Π_1 does not admit a polynomial kernel, then neither does Π_2 [3]. Next, let us define the Red Blue Dominating Set problem. The input is a bipartite graph G with bipartition (A, B) and an integer t, this problem asks if B has a subset of at most t vertices that dominates A. This problem is NP-complete [11] and it does not have a polynomial kernel when parameterized by $|A|$ [8]. It was shown that Tree Contraction and Star Contraction do not admit a polynomial kernel, by a polynomial parameter preserving reduction from this problem [16]. We build upon the reductions in [16] to show that the two remaining problems also do not admit a polynomial kernel. The following theorem is the main result of this paper.

▶ **Theorem 1.1.** *Given a graph (digraph) G on n vertices, an integer k and an approximation parameter $\alpha > 1$, there is an algorithm that runs in $k^{f(\alpha)}n^{\mathcal{O}(1)}$ time and outputs a graph (digraph) G' on $k^{g(\alpha)}$ vertices and an integer k' such that for every $c > 1$, a c-approximate (tree/star/cactus/out-tree contraction) solution for (G', k') can be turned into a $(c\alpha)$-approximate (tree/star/cactus/out-tree contraction) solution for (G, k) in $n^{\mathcal{O}(1)}$. Here $f(\alpha)$ and $g(\alpha)$ are constants depending on α.*

2 Preliminaries

An undirected graph is a pair consisting of a set V of vertices and a set E of edges where $E \subseteq V \times V$. An edge is specified as an unordered pair of vertices. For a graph G, $V(G)$ and $E(G)$ denote the set of vertices and edges respectively. Two vertices u, v are said to be *adjacent* if there is an edge uv in the graph. The neighbourhood of a vertex v, denoted by $N_G(v)$, is the set of vertices adjacent to v and its degree $d_G(v)$ is $|N_G(v)|$. The subscript in the notation for neighbourhood and degree is omitted if the graph under consideration is clear. For a set of edges F, $V(F)$ denotes the set of endpoints of edges in F. For a set $S \subseteq V(G)$, $G - S$ denotes the graph obtained by deleting S from G and $G[S]$ denotes the subgraph of G induced on set S. For graph theoretic terms and notation which are not explicitly defined here, we refer the reader to the book by Diestel [7].

Two non-adjacent vertices u and v are called as *false twins* of each other if $N(u) = N(v)$. A *path* $P = (v_1, \ldots, v_l)$ is a sequence of distinct vertices where every consecutive pair of vertices is adjacent. The vertices of P is the set $\{v_1, \ldots, v_l\}$ and is denoted by $V(P)$. The *length* of a path is $|V(P)| - 1$. A *cycle* is a sequence $(v_1, \ldots, v_l, v_1)$ of vertices such that $(v_1, \ldots, v_l)$ is a path and $v_l v_1$ is an edge. A *leaf* is a vertex of degree 1. A graph is called *connected* if there is a path between any pair of its vertices and it is called *disconnected* otherwise. A *cut vertex* of a connected graph G is a vertex v such that $G - \{v\}$ is disconnected. A graph that has no cut vertex is called *2-connected*. A *component* of a disconnected graph is a maximal connected subgraph. A set $S \subseteq V(G)$ is called a *vertex cover* if for every edge uv, either $u \in S$ or $v \in S$. Further, S is called a *connected vertex cover* if $G[S]$ is connected. A set $I \subseteq V(G)$ of pairwise non-adjacent vertices is called as an *independent set*. A set S of vertices is said to *dominate* another set S' of vertices if for every vertex v in S', $N(v) \cap S \neq \emptyset$. A *tree* is a connected acyclic graph. A *star* is a tree in which there is a path of length at most 2 between any 2 vertices. A graph is called a *cactus* if every edge is a part of at most one cycle.

The *contraction* operation of an edge $e = uv$ in G results in the deletion of u and v and the addition of a new vertex w adjacent to vertices that were adjacent to either u or v. Any parallel edges added in the process are deleted so that the graph remains simple. The resulting graph is denoted by G/e. Formally, $V(G/e) = V(G) \cup \{w\} \setminus \{u, v\}$ and $E(G/e) = \{xy \mid x, y \in V(G) \setminus \{u, v\}, xy \in E(G)\} \cup \{wx \mid x \in N_G(u) \cup N_G(v)\}$. For a set of edges $F \subseteq E(G)$, G/F denotes the graph obtained from G by contracting the edges in F in an arbitrary order. It is easy to see that G/F is oblivious to the contraction sequence. A graph G is *contractible* to a graph T, if T can be obtained from G by a sequence of edge contractions. For graphs G and T with $V(T) = \{t_1, \cdots, t_l\}$, G is said to have a *T-witness structure* $\mathcal{W}$ if $\mathcal{W}$ is a partition of $V(G)$ into l sets and there is a bijection $W : V(T) \mapsto \mathcal{W}$ such that the following properties hold.

- For each $t_i \in V(T)$, $G[W(t_i)]$ is connected.
- For a pair $t_i, t_j \in V(T)$, $t_i t_j \in E(T)$ if and only if there is an edge between a vertex in $W(t_i)$ and a vertex in $W(t_j)$ in G.

The sets $W(t_1), \cdots W(t_l)$ in $\mathcal{W}$ are called *witness sets*. It is easy to observe the following (also see [16]).

▶ **Observation 1.** G is contractible to T if and only if G has a T-witness structure.

We associate a T-witness structure $\mathcal{W}$ of G with a set $F \subseteq E(G)$ whose contraction in G results in T, by defining F to be the union of the set of edges of a spanning tree of $G[W]$, for each $W \in \mathcal{W}$. Note that there is a unique T-witness structure of G corresponding to a set F.

▶ **Observation 2.** $|F| = \sum\limits_{W \in \mathcal{W}} (|W| - 1)$.

We say that, G is said to be $|F|$-contractible to T and it is easy to verify the following. For every $W \in \mathcal{W}$, $|W| \leq |F| + 1$, and further, $|\{W \in \mathcal{W} \mid |W| > 1\}| \leq |F|$. Finally, we have the following observation on the neighbors of vertices in $W(t)$, when t is a leaf in T.

▶ **Observation 3.** Let t be a leaf in T and t' be its unique neighbour. Then, $\bigcup_{v \in W(t)} N_G(v) \subseteq W(t') \cup W(t)$.

Proof. Consider a leaf t in T. Assume on the contrary that there exists t' and t'' (distinct from t) such that $N(u) \cap W(t') \neq \emptyset$ and $N(v) \cap W(t'') \neq \emptyset$ for some u and v (not necessarily distinct) in $W(t)$. Then, t has degree at least 2 contradicting the fact that it is a leaf. ◀

We denote the set of integers from 1 to n by $[n]$. We also use the bound, $\frac{x+p}{y+q} \leq \max\{\frac{x}{y}, \frac{p}{q}\}$ for any positive real numbers x, y, p, q, to prove that the reduction rules we define are strict α-approximate for some real number α. In this paper, use TC$(\cdot)$, OTC$(\cdot)$ and CC$(\cdot)$ to denote the parameterized minimization version of Tree Contraction, Out-Tree Contraction and Cactus Contraction, respectively. Recall that these functions assign a value of $k+1$ to any solution of value $k+1$ or more.

3 Tree Contraction

We begin with the Tree Contraction problem, which admits a $4^k n^{\mathcal{O}(1)}$ algorithm (where n is the number of vertices of the input graph) by using a FPT algorithm for Connected Vertex Cover as a subroutine, and further it does not admit a polynomial kernel unless NP $\subseteq$ coNP/*poly* [16]. This lower-bound also holds for Star Contraction. Before we proceed to describe a PSAKS for these problems, we mention the following simplifying assumption known from [16] which states that, the tree witness structure of a graph can be constructed from the tree witness structures of its 2-connected components.

▶ **Lemma 3.1** ([16]). *A connected graph is k-contractible to a tree if and only if each of its 2-connected components is contractible to a tree using at most k edge contractions in total.*

Observe that there can be at most k non-trivial 2-connected components in the input graph, and we can consider each such component separately. The output of our kernelization algorithm will be a disjoint union of the kernels for each 2-connected component. So from now onwards we assume that the input graph is 2-connected. Next, we make some observations on the tree witness structure of a graph.

▶ **Lemma 3.2.** *Let F be a minimal set of edges of a 2-connected graph G such that G/F is a tree T with $V(T) = \{t_1, t_2, \ldots, t_l\}$ and $l \geq 3$. Let $\mathcal{W}$ denote the corresponding T-witness structure of G. Then there exists a set F' of at most $|F|$ edges of G such that, G/F' is a tree T' and the corresponding T'-witness structure $\mathcal{W}'$ of G satisfies the following property: $W'(t') \in \mathcal{W}'$ is a singleton set, if and only if t' is a leaf in T'.*

Proof. First, we show that every vertex $t \in V(T)$ such that $|W(t)| = 1$ is a leaf in T. Suppose there is a non-leaf t in T such that $W(t) = \{u\}$ for some $u \in V(G)$. Then, $T - \{t\}$ has at least two non-empty subtrees, say T_1 and T_2. Consider $U_1 = \bigcup_{t \in V(T_1)} W(t)$ and $U_2 = \bigcup_{t \in V(T_2)} W(t)$. As $\mathcal{W}$ is the corresponding T-witness structure of G, it follows that there is no edge between a vertex in U_1 and a vertex in U_2 in $G - \{u\}$. This contradicts the fact that G is 2-connected.

Now, consider a leaf t_i in T such that $|W(t_i)| > 1$. Let t_j be the unique neighbour of t_i, and note that t_j is not a leaf in T. As $t_i t_j \in E(T)$, there exists an edge in G between a vertex in $W(t_i)$ and a vertex in $W(t_j)$. Therefore, $G[W(t_i) \cup W(t_j)]$ is connected. We claim that $G[W(t_i) \cup W(t_j)]$ has a spanning tree which has a leaf from $W(t_i)$. Observe that as $|W(t_i)| > 1$, any spanning tree of $G[W(t_i)]$ has at least 2 leaves. If there is a spanning tree of $G[W(t_i)]$ that has a leaf u which is not adjacent to any vertex in $W(t_j)$, then $G[(W(t_i) \cup W(t_j)) \setminus \{u\}]$ is connected too and u is the required vertex. Otherwise, every leaf in every spanning tree of $G[W(t_i)]$ is adjacent to some vertex in $W(t_j)$ and hence $G[(W(t_i) \cup W(t_j)) \setminus \{u\}]$ is connected for each vertex $u \in W(t_i)$. Therefore, as claimed, $G[W(t_i) \cup W(t_j)]$ has a spanning tree which has a leaf v from $W(t_i)$. Consider the partition $\mathcal{W}' = (\mathcal{W} \cup \{W_v, W_{ij}\}) \setminus \{W(t_i), W(t_j)\}$ of G where $W_v = \{v\}$ and $W_{ij} = (W(t_j) \cup W(t_i)) \setminus \{v\}$. Then, as $N(v) \subseteq W(t_i) \cup W(t_j)$ by Observation 3, it follows that $\mathcal{W}'$ is a T'-witness structure of G, where T' is a tree. Further,

T' is the tree obtained from T by adding a new vertex t_{ij} adjacent to $N(t_j)$ and a new vertex t_v adjacent to t_{ij} and then deleting t_i, t_j. This leads to a set F' of at most $|F|$ edges of G such that $T' = G/F'$ is a tree. Further note that, T and T' has the same number of vertices. Now recall that T has at least 3 vertices and hence it has at least one non-leaf vertex. Further, every leaf vertex is adjacent to some non-leaf vertex in T (and T'). Therefore we repeat the above process for every non-singleton leaf, which proves the lemma. ◀

Let us remark that, it is safe to assume that T has at least 3 vertices. Otherwise, we can bound the number of vertices in G by $|F| + 2$, and since we are concerned with solutions of value k or smaller, this gives us a trivial kernel. Indeed, the main challenge lies in bounding the set of singleton witness sets. Subsequently, we assume that all tree witness structures have this property. Lemma 3.2 immediately leads to the following equivalence of Star Contraction and Connected Vertex Cover.

▶ **Lemma 3.3** ($\star$[1]). *G has a set $F \subseteq E(G)$ such that G/F is a star if and only if G has a connected vertex cover of size $|F| + 1$.*

As Connected Vertex Cover has a PSAKS [18], we have the following result.

▶ **Theorem 3.4.** Star Contraction *parameterized by the solution size admits a PSAKS.*

Lemma 3.2 also leads to the following relationship between Tree Contraction and Connected Vertex Cover.

▶ **Lemma 3.5.** *If G is k-contractible to a tree, then G has a connected vertex cover of size at most $2k$.*

Proof. As G is k-contractible to a tree, there exists a (minimal) set of edges F such that $|F| \leq k$ and $T = G/F$ is a tree. Let $\mathcal{W}$ be the corresponding T-witness structure of G and $\mathcal{W}'$ denote the set of non-singleton sets in $\mathcal{W}$. Let X denote the set of vertices of G which are in a set in $\mathcal{W}'$. By Lemma 3.2, we can assume that every leaf of T corresponds to a singleton witness set, and vice versa. Let L be the set of leaves of T. Then, $I = \{v \in V(G) \mid v \in W(t), t \in L\}$ is an independent set in G. Thus, X is a vertex cover of G. As $|F| \leq k$, we have $|X| \leq 2k$ as every vertex in X has an edge incident on it that is in F. Finally, since the set of non-leaves of a tree induces a subtree, it follows that $G[X]$ is connected. ◀

Now, we move on to describe a PSAKS for Tree Contraction. We define a partition of vertices of G into the following three parts:

$$H = \{u \in V(G) \mid d(u) \geq 2k+1\},$$
$$I = \{v \in V(G) \setminus H \mid N(v) \subseteq H\}.$$
$$R = V(G) \setminus (H \cup I).$$

We define the first reduction rule as follows.

▶ **Reduction Rule 3.1.** If there is a vertex $v \in I$ that has at least $2k + 1$ false twins, then delete v. That is, the resultant instance is $(G - \{v\}, k)$.

▶ **Lemma 3.6.** *Reduction Rule 3.1 is safe.*

[1] Proofs of results marked $\star$ have been omitted due to the lack of space. They will appear in the full version of the paper.

Proof. Consider a solution F' of the reduced instance (G', k'). If $|F'| \geq k' + 1$, then the solution lifting algorithm returns $E(G)$, otherwise it returns $F = F'$. We show that this solution lifting algorithm with the reduction rule constitutes a strict 1-approximate polynomial time preprocessing algorithm. If $|F'| \geq k' + 1$ then $\text{TC}(G, k, F) \leq k + 1 = \text{TC}(G', k', F')$. Otherwise, $|F'| \leq k$ and let T' be the tree G'/F' and $\mathcal{W}'$ denote the corresponding T'-witness structure of G'. Then, as v has at least $2k + 1$ false twins, one of these twins, say u, is not in $V(F')$. In other words, there is a vertex t in T' such that $W'(t) = \{u\}$. By Lemma 3.2, t is a leaf. Let t' denote the unique neighbour of t in T'. Then, from Observation 3, $N_{G'}(u) \subseteq W'(t')$. Let T be the tree obtained from T' by adding a new vertex t_v as a leaf adjacent to t'. Since $N_{G'}(u) = N_G(u) = N_G(v)$, all the vertices in $N_G(v)$ are in $W'(t')$. Define the partition $\mathcal{W}$ of $V(G)$ obtained from $\mathcal{W}'$ by adding a new set $\{v\}$. Then, G/F is T and $\mathcal{W}$ is the corresponding T-witness structure of G. Hence, $\text{TC}(G, k, F) \leq \text{TC}(G', k', F')$.

Next, consider an optimum solution F^* for (G, k). If $|F^*| \geq k + 1$ then $\text{OPT}(G, k) = k + 1 \geq \text{OPT}(G', k')$. Otherwise, $|F^*| \leq k$ and let $T = G/F^*$. Let $\mathcal{W}^*$ denote the corresponding T-witness structure of G. If there is a leaf t in T such that $W^*(t) = \{v\}$, then F^* is also a solution for (G', k') and $\text{OPT}(G', k') \leq \text{OPT}(G, k)$. Otherwise, as v has at least $2k + 1$ false twins, one of these twins, say u, is not in $V(F^*)$. That is, there is a leaf t in T such that $W^*(t) = \{u\}$. Define the partition $\mathcal{W}'$ of $V(G)$ obtained from $\mathcal{W}^*$ by replacing u by v and v by u. Then, the set F' of edges of G obtained from F by replacing the edge xv with the edge xu for each x is also an optimum solution for (G, k). Further, it is a solution for (G', k') and therefore, $\text{OPT}(G', k') \leq \text{OPT}(G, k)$. Hence, $\frac{\text{TC}(G,k,F)}{\text{OPT}(G,k)} \leq \frac{\text{TC}(G',k',F')}{\text{OPT}(G',k')}$. ◀

Given $\alpha > 1$, let d be the minimum integer such that $\alpha \geq \frac{d}{d-1}$. That is, $d = \lceil \frac{\alpha}{\alpha - 1} \rceil$. The second reduction rule is the following.

▶ **Reduction Rule 3.2.** If there are vertices $v_1, v_2, \ldots, v_{2k+1} \in I$ and $h_1, h_2, \ldots, h_d \in H$ such that $\{h_1, \ldots, h_d\} \subseteq N(v_i)$ for each $i \in [2k + 1]$ then contract all edges in $\tilde{E} = \{v_1 h_i \mid i \in [d]\}$ and reduce the parameter by $d - 1$. The resulting instance is $(G/\tilde{E}, k - d + 1)$.

▶ **Lemma 3.7.** *Reduction Rule 3.2 is α-safe.*

Proof. Consider a solution F' of the reduced instance (G', k'). If $|F'| \geq k' + 1$, then the solution lifting algorithm returns $E(G)$, otherwise it returns $F = F' \cup \tilde{E}$. We will show that this solution lifting algorithm with the reduction rule constitutes a strict α-approximate polynomial time preprocessing algorithm. First, we prove that $\text{TC}(G, k, F) \leq \text{TC}(G', k', F') + d$. If $|F'| \geq k' + 1$ then $\text{TC}(G', k', F') = k' + 1$. In this case, $F = E(G)$ and $\text{TC}(G, k, F) \leq k + 1 = k' + d = \text{TC}(G', k', F') + d - 1$. Consider the case when $|F'| \leq k'$ and let $\mathcal{W}' = \{W'(t_1), W'(t_2), \ldots, W'(t_l)\}$ be the corresponding G'/F'-witness structure of G. Let w denote the vertex in $V(G') \setminus V(G)$ obtained by contracting $\tilde{E}$. Without loss of generality, assume that $w \in W'(t_1)$. Define $\mathcal{W} = (\mathcal{W}' \cup \{W_1\}) \setminus \{W'(t_1)\}$ where $W_1 = (W'(t_1) \cup \{v_1, h_1, h_2, \ldots, h_d\}) \setminus \{w\}$. Note that $V(G) \setminus \{v_1, h_1, h_2, \ldots, h_d\} = V(G') \setminus \{w\}$ and hence $\mathcal{W}$ is partition of $V(G)$. Further, $G[W_1]$ is connected as $G'[W'(t_1)]$ is connected. A spanning tree of $G'[W'(t_1)]$ along with $\tilde{E}$ is a spanning tree of $G[W_1]$. Also, $|W_1| = |W'(t_1)| + d$ and any vertex which is adjacent to w in G' is adjacent to at least one vertex in $\{v_1, h_1, h_2, \ldots, h_d\}$ in G. Thus, $\mathcal{W}$ is a G/F-witness structure of G where G/F is a tree isomorphic to G'/F'. Therefore, $\text{TC}(G, k, F) \leq \text{TC}(G', k', F') + d$.

We now show that $\text{OPT}(G', k') \leq \text{OPT}(G, k) - (d - 1)$. Let F^* be an optimum solution for (G, k) and $\mathcal{W}$ be the corresponding G/F^*-witness structure of G. Let T be G/F^*. If $|F^*| \geq k + 1$, then $\text{OPT}(G, k) = k + 1 = k' + d \geq \text{OPT}(G', k') + d - 1$. Otherwise, $|F^*| \leq k$ and there is at least one vertex, say v_q in $\{v_1, v_2, \ldots, v_{2k+1}\}$ which is not in

$V(F^*)$. By Observation 3, $N(v_q)$ and hence $\{h_1, h_2, \ldots, h_d\}$ are in the same witness set, say $W(t_i)$ where $t_i \in V(T)$. If $v_1 \in W(t_i)$ then $F' = F^* \setminus \tilde{E}$ is solution to (G', k') and so $\text{OPT}(G', k') \leq |F'| = |F^*| - d = \text{OPT}(G, k) - d$. Otherwise, $v_1 \notin W(t_i)$ and let $t_j \in V(T)$ be the vertex such that $v_1 \in W(t_j)$. Then, t_i and t_j are adjacent in T. Define another partition $\mathcal{W}' = \mathcal{W} \cup \{W(t_{ij})\} \setminus \{W(t_i), W(t_j)\}$ of $V(G)$ where $W(t_{ij}) = W(t_i) \cup W(t_j)$. Clearly, $G[W(t_{ij})]$ is connected. Thus, $\mathcal{W}'$ is a G/F-witness structure of G where $|F| = |F^*| + 1$ as $|W(t_i)| - 1 + |W(t_j)| - 1 = (|W(t_{ij})| - 1) - 1$. In particular, G/F is the tree obtained from G/F^* by contracting the edge $t_i t_j$. Finally, without loss of generality $\tilde{E} \subseteq F$ and thus $F' = F \setminus \tilde{E}$ is a solution to (G', k'). Therefore, $\text{OPT}(G', k') \leq |F'| = |F^*| + 1 - d = \text{OPT}(G, k) - d + 1$. Combining these bounds, we have $\frac{\text{TC}(G,k,F)}{\text{OPT}(G,k)} \leq \frac{\text{TC}(G',k',F')+d}{\text{OPT}(G',k')+(d-1)} \leq \max\left\{\frac{\text{TC}(G',k',F')}{\text{OPT}(G',k')}, \alpha\right\}$. ◀

It is easy to see that the above rule can be applied in $\mathcal{O}((2k)^d \cdot n^c)$ time, by considering each subset of H of cardinality d, where c is a constant independent of α and n is the number of vertices in the graph. This leads to the following bound.

▶ **Lemma 3.8.** *Suppose G is k-contractible to a tree and neither of the Reduction rules 3.1 and 3.2 are applicable on the instance (G, k). Then, $|V(G)|$ is $\mathcal{O}((2k)^{d+1} + k^2)$.*

Proof. We will bound H, I and R separately in order to bound $V(G)$. By Lemma 3.5, G has a connected vertex cover S of size at most $2k$. As H is the set of vertices of degree at least $2k+1$, $H \subseteq S$ and so $|H| \leq 2k$. Every vertex in R has degree at most $2k$. Therefore, as $S \cap R$ is a vertex cover of $G[R]$, $|E(G[R])|$ is $\mathcal{O}(k^2)$. Also, by the definition of I, every vertex in R has a neighbour in R and hence there are no isolated vertices in $G[R]$. Thus, $|R|$ is $\mathcal{O}(k^2)$. Finally, we bound the size of I. For every set $H' \subseteq H$ of cardinality less than d, there are at most $2k+1$ vertices in I which have H' as their neighbourhood. Otherwise, Reduction Rule 3.1 would have been applied. Hence, there are at most $(2k+1) \cdot \binom{2k}{d-1}$ vertices in I which have degree less than d. Further, for a d-size subset H' of H, there are at most $2k+1$ vertices in I which contain H' in their neighbourhood. Otherwise, Reduction Rule 3.2 would have been applied. As a vertex in I of degree at least d is adjacent to all vertices in at least one such subset of H, there are at most $(2k+1)\binom{2k}{d}$ vertices of I of degree at least d. Therefore, $|I|$ is $\mathcal{O}((2k)^{d+1})$. ◀

Now, we have a PSAKS for the problem.

▶ **Theorem 3.9** ($\star$). *Tree Contraction admits a strict PSAKS with $\mathcal{O}((2k)^{\lceil \frac{\alpha}{\alpha-1} \rceil + 2} + k^3)$ vertices.*

4 Out-Tree Contraction

In this section, we describe a PSAKS for an analogue of Tree Contraction in directed graphs. We first require some terminology on directed graphs. A *directed graph (or digraph)* is a pair consisting of a set V of vertices and a set A of directed edges (arcs) where $A \subseteq V \times V$. An arc is specified as an ordered pair of vertices uv and we say that the arc uv is directed from u to v. Let $V(D)$ and $A(D)$ denote the sets of vertices and arcs of a digraph D. For a vertex $v \in V(D)$, $N^-(v)$ denotes the set $\{u \in V(D) \mid uv \in A(D)\}$ of its in-neighbors and $N^+(v)$ denotes the set $\{u \in V(D) \mid vu \in A(D)\}$ of its out-neighbors. The neighborhood of a vertex v is the set $N(v) = N^+(v) \cup N^-(v)$. The in-degree of a vertex v, denoted by $d^-(v)$, is $|N^-(v)|$. Similarly, its out-degree is $|N^+(v)|$ which is denoted by $d^+(v)$. The (total) degree of v, denoted by $d(v)$, is the sum of its in-degree and out-degree. A sequence $P = (v_1, \cdots, v_l)$ of distinct vertices of D is called a directed path in D if $v_1v_2, \cdots, v_{l-1}v_l \in A(D)$.

For a digraph D, its underlying undirected graph G_D is the undirected graph on the vertex set $V(D)$ with the edge set $\{uv \mid uv \in A(D)\}$. An out-tree T is a digraph where each vertex has in-degree at most 1 such that G_T is a tree. A vertex v of an out-tree is called a *leaf* if $d^-(v) = 1$ and $d^+(v) = 0$. The *root* of an out-tree is the unique vertex that has no in-neighbor. The contraction of an arc $e = uv$ in D results in the digraph, denoted by D/e, on the vertex set $V' = V(D) \setminus \{u, v\} \cup \{x\}$ with $A(D/e) = \{pq \mid pq \in A(D) \text{ and } p, q \in V'\} \cup \{xz \mid vz \in A(D)\} \cup \{zx \mid zu \in A(D)\} \cup \{xz \mid uz \in A(D)\} \cup \{zx \mid zv \in A(D)\}$. The notion of witness structures and witness sets are extended to digraphs as follows. For digraphs D and T with $V(T) = \{t_1, \cdots, t_l\}$, D is said to have a T-witness structure $\mathcal{W}$ if $\mathcal{W}$ is a partition of $V(D)$ into l sets (called witness sets) and there is a bijection $W : V(T) \mapsto \mathcal{W}$ such that the following properties hold.

- For each $t_i \in V(T)$, $G_D[W(t_i)]$ is connected.
- For a pair $t_i, t_j \in V(T)$, $t_it_j \in A(T)$ if and only if there is an arc from a vertex in $W(t_i)$ to a vertex in $W(t_j)$ in D.

Analogous to undirected graphs, we associate a T-witness structure $\mathcal{W}$ of G with a set $F \subseteq A(D)$ whose contraction in D results in T by defining F to be the set of the arcs corresponding to the edges of a spanning tree of $G_D[W]$ for each $W \in \mathcal{W}$. Now, we show that similar to Tree Contraction, Out-Tree Contraction also does not admit a polynomial kernel. We modify the reduction known for Tree Contraction to show this hardness.

▶ **Lemma 4.1** ($\star$). Out-Tree Contraction *does not have a polynomial kernel unless* NP $\subseteq$ coNP*/poly.*

Now, we describe a PSAKS for Out-Tree Contraction. We note that the simplifying assumptions in Tree Contraction, such as ignoring cut vertices and requiring that the leaves of the resultant tree correspond to singleton witness sets, do not hold anymore. Our first reduction rule is based on the observation that the digraph obtained from an out-tree, by adding a new vertex as an out-neighbor of any vertex, is once again an out-tree.

▶ **Reduction Rule 4.1.** If there is a vertex $v \in V(D)$ with $d^-(v) = 1$ and $d^+(v) = 0$ then delete v. The resulting instance is (D', k') where $D' = D - \{v\}$ and $k' = k$.

▶ **Lemma 4.2** ($\star$). *Reduction Rule 4.1 is safe.*

The operation of subdividing an arc uv in D results in the deletion of the arc uv and the addition of a new vertex w as an out-neighbor of u and an in-neighbor of v. The next reduction rule is based on the observation that subdividing an arc of an out-tree results in another out-tree. To exploit this observation, we need the following lemma.

▶ **Lemma 4.3** ($\star$). *Suppose D has a directed path $P = (v_0, v_1, \dots, v_l, v_{l+1})$ with $l > k + 1$ and $d^-(v) = d^+(v) = 1$ for each $v \in V(P)$. Then, no minimal out-tree contraction solution F of D with $|F| \leq k$ contains an arc incident on $V(P) \setminus \{v_0, v_{l+1}\}$.*

▶ **Reduction Rule 4.2.** If there is a directed path $P = (v_0, v_1, \dots, v_l, v_{l+1})$ with $l > k+2$ and $d^-(v) = d^+(v) = 1$ for each $v \in V(P)$, then replace P by the path $P' = (v_0, v_1, \dots, v_{k+2}, v_{l+1})$. Specifically, the resulting instance is $(D', k' = k)$ where D' is the digraph obtained from D by deleting $\{v_{k+3}, \dots, v_l\}$ and adding the arc $v_{k+2}v_{l+1}$.

We note that this rule can be applied in polynomial time by searching for such a path in the subgraph induced on the vertices of degree 2.

▶ **Lemma 4.4** ($\star$). *Reduction Rule 4.2 is safe.*

Before we describe the next reduction rule, we define the following partition of $V(D)$:

$$I = \{v \in V(D) \mid d^+(v) = 0\},$$
$$H = V(D) \setminus I.$$

Now, we apply the following reduction rule on I.

▶ **Reduction Rule 4.3.** If there are vertices $v, v_1, v_2, \ldots, v_{2k+1} \in I$ such that $N^-(v) = N^-(v_1) = \cdots = N^-(v_{2k+1})$, then delete v. The resulting instance is (D', k') where $D' = D - \{v\}$ and $k' = k$.

▶ **Lemma 4.5** ($\star$). *Reduction Rule 4.3 is safe.*

Now, we describe the final reduction rule. Given $\alpha > 1$, let d be the minimum integer such that $\alpha \geq \frac{d}{d-1}$.

▶ **Reduction Rule 4.4.** If there are vertices $v_1, v_2, \ldots, v_{2k+1} \in I$ and $h_1, h_2, \ldots, h_d \in H$ such that $\{h_1, \ldots, h_d\} \subseteq N(v_i)$ for each $i \in [2k+1]$, then contract arcs in $\tilde{A} = \{(h_i v_1) \mid i \in [d]\}$ and reduce the parameter by $d-1$. That is, the resulting instance is $(D/\tilde{A}, k-(d-1))$.

▶ **Lemma 4.6** ($\star$). *Reduction Rule 4.4 is α-safe.*

Now, we prove that the reduction rules described lead to a lossy kernel of polynomial size.

▶ **Lemma 4.7** ($\star$). *Suppose D is k-contractible to an out-tree and none of Reduction Rules 4.1,4.2,4.3 and 4.4 are applicable on the instance (D, k). Then, $|V(D)|$ is $\mathcal{O}((2k)^{d+1} + k^2)$.*

Now, we have the following result.

▶ **Theorem 4.8** ($\star$). Out-Tree Contraction *admits a PSAKS with $\mathcal{O}(k^{2\lceil\frac{\alpha}{\alpha-1}\rceil+1} + k^2)$ vertices.*

5 Cactus Contraction

As mentioned earlier, Tree Contraction has been shown not to admit a polynomial kernel unless NP $\subseteq$ coNP/*poly* by a reduction from Red Blue Dominating Set [16]. We modify this reduction to show similar hardness for Cactus Contraction.

▶ **Lemma 5.1** ($\star$). Cactus Contraction *does not have a polynomial kernel unless* NP$\subseteq$ coNP*/poly.*

Next, we proceed to describe a PSAKS for Cactus Contraction. We first list the following simplifying assumption.

▶ **Lemma 5.2** ($\star$). *A connected graph is k-contractible to a cactus if and only if each of its 2-connected components is contractible to a cactus using at most k edge contractions in total.*

So, without loss of generality, we assume that the input graph G is 2-connected. Before we proceed to describe the reduction rules, we need to define some additional terminology. The operation of *subdividing* an edge uv results in the graph obtained by deleting uv and adding a new vertex w adjacent to both u and v. The operation of *short-circuiting* a degree 2 vertex v with neighbors u and w results in the graph obtained by deleting v and then adding the edge uw if it is not already present.

▶ **Observation 4.** The following statements hold for a cactus T.

1. Every vertex of degree at least 3 in T is a cut vertex.
2. The graph obtained by subdividing an edge of T is a cactus.
3. The graph obtained by short-circuiting a degree 2 vertex v in T is a cactus.

Next, we will make some observations on a cactus witness structure of a graph. We define the following terms, which are useful for the forthcoming results. For a path P in G, let $N(P)$ denote the neighborhood of P, i.e. the set of vertices in $V(G) \setminus V(P)$ that are adjacent to a vertex in P.

▶ **Definition 5.3** (Simple Path)**.** A path P in a graph G is called simple path of G, if every internal vertex of P has degree exactly equal to two in G. Observe that, the endpoints of the path P are the only vertices with a neighbor in $G \setminus P$.

We say that a simple path P is neighbor to a set of vertices W disjoint from $V(P)$, if each vertex in W is neighbor of at least one of the two endpoints of P. We denote the set of all neighbors of a simple path P in G by $N(P)$.

▶ **Definition 5.4** (Pendent Cycle)**.** For a cactus T, a cycle C is called an pendent cycle if it contains exactly one cut vertex.

Let T be a cactus obtained by contracting a set of edges in the graph G with a witness structure $\mathcal{W}$. Now, consider a pendent cycle (uPu) in cactus T where u is the cut vertex, and for every $t \in P$, $|W(t)| = 1$. Then observe that P corresponds to a simple path in G, and with a slight abuse of notation let us use P to denote the path in G as well. Observe the set of internal vertices of any two simple paths are disjoint. We say that a simple path P in G forms a pendent cycle in T if there is a cut vertex u in T such that uPu is a pendent cycle in T. The following lemma establishes a few more properties of the cactus witness structure.

▶ **Lemma 5.5** ($\star$)**.** *Let F be a minimal set of edges of a 2-connected graph G such that G/F is a cactus T with $V(T) = \{t_1, t_2, \dots, t_l\}$ and $l \geq 3$. Let $\mathcal{W}$ be the corresponding T-witness structure of G. Then the following properties hold.*

1. *There exists a set F' of at most $|F|$ edges of G such that G/F' is a cactus and the corresponding G/F'-witness structure $\mathcal{W}'$ of G satisfies the property that for every leaf t in G/F', $W'(t) \in \mathcal{W}'$ is a singleton set.*
2. *For any three vertices u_1, u_2 and u_3 such that, $W(t_1) = \{u_1\}, W(t_2) = \{u_2\}, W(t_3) = \{u_3\}$, there is a vertex $t \in V(T)$ such that $(N(u_1) \cap N(u_2) \cap N(u_3)) \subseteq W(t)$. Similarly, for any three simple paths P_1, P_2 and P_3 in G, such that each of them form a pendent cycle in T, there is a vertex $t \in V(T)$ such that $N_G(P_1) \cap N_G(P_2) \cap N_G(P_3) \subseteq W(t)$.*
3. *If t is cut vertex in T then $|W(t)| > 1$.*
4. *If $|F| \leq k$ and $d(v) \geq k+3$, then there is a vertex $t \in V(T)$ such that $v \in W(t)$ and $|W(t)| > 1$.*

Let us remark that, it is safe to assume that T has at least 3 vertices, as otherwise T is just an edge. Therefore, we can bound the number of vertices in G by $|F| + 2$, and since we are concerned with solutions of value k or smaller, this gives us a trivial kernel. Indeed, the main challenge lies in bounding the set of singleton witness sets. Subsequently, we assume that all cactus witness structures have these properties.

▶ **Lemma 5.6** ($\star$)**.** *Suppose G has a path $P = (u_0, u_1, \dots, u_l, u_{l+1})$ with $l > k+1$ consisting of vertices of degree 2. Then, no minimal cactus contraction solution F of G with $|F| \leq k$ contains an edge incident on $V(P) \setminus \{u_0, u_{l+1}\}$.*

Now, we are ready to state the first reduction rule.

▶ **Reduction Rule 5.1.** If G has a path $P = (u_0, u_1, \ldots, u_l, u_{l+1})$ such that $l > k+2$ consisting of vertices of degree 2, then replace P by the path $P' = (u_0, u_1, \ldots, u_{k+2}, u_{l+1})$. In other words, the resulting instance is $(G', k' = k)$ where G' is the graph obtained from G by deleting $\{u_{k+3}, \ldots, u_l\}$ and adding the edge $u_{k+2}u_{l+1}$.

We observe that this rule can be applied in polynomial time by considering each simple path in the graph of length more than $k+1$.

▶ **Lemma 5.7** ($\star$)**.** *Reduction Rule 5.1 is safe.*

We apply Reduction Rule 5.1 exhaustively to the graph G, and observe that any simple path contains at most $k+4$ vertices. Now, we partition $V(G)$ into the following four parts:

$$\begin{aligned}
H &= \{u \in V(G) \mid d(u) \geq k+3\},\\
I_v &= \{v \in V(G) \setminus H \mid N_G(v) \subseteq H\},\\
I_p &= \{V(P) \mid P \text{ is a simple path in } G \text{ and } N_G(P) \subseteq H\},\\
R &= V(G) \setminus (H \cup I_v \cup I_p).
\end{aligned}$$

With a slight abuse of notation, we say that a path P is contained in I_p (i.e. $P \in I_p$) if $V(P) \subseteq I_P$. Let us make the following observation.

▶ **Observation 5.** For any two paths $P_1, P_2 \in I_p$, we have $V(P_1) \cap V(P_2) = \emptyset$.

▶ **Lemma 5.8** ($\star$)**.** *Let G is k-contractible to a cactus such that Reduction Rule 5.1 is not applicable on (G, k). If H, R are the partitions defined above, then $|H \cup R|$ is at most $\mathcal{O}(k^4)$.*

For our next reduction rule, we extend the notion of false twins to simple paths. We call two paths, P_1 and P_2 in I_p, *false twins* if $N(P_1) = N(P_2)$.

▶ **Reduction Rule 5.2.** If there is a vertex $v \in I_v$ that has at least $2k+3$ false twins, then delete v. That is, the resultant instance is $(G - \{v\}, k)$. Similarly, if there is a path P in I_p that has at least $2k+3$ false twins, then delete P.

▶ **Lemma 5.9** ($\star$)**.** *Reduction Rule 5.2 is safe.*

Given $\alpha > 1$, let d be $\lceil \frac{\alpha}{\alpha-1} \rceil$. For every simple path $P \in I_p$, such that $N(P)$ contains at least $2d$ vertices of H, pick one of its endpoints, that is adjacent to at least d vertices of H, into the set $\tilde{I}_p$ We apply the following reduction rule to the set $I = I_v \cup \tilde{I}_p$.

▶ **Reduction Rule 5.3.** If there are vertices $v_1, v_2, \ldots, v_{2k+3} \in I$ and $h_1, h_2, \ldots, h_d \in H$ such that $\{h_1, \ldots, h_d\} \subseteq N(v_i)$ for all $i \in [2k+3]$ then contract all edges in $\tilde{E} = \{v_1h_i \mid i \in [d]\}$ and reduce the parameter by $d-1$. The resulting instance is $(G/\tilde{E}, k-d+1)$.

▶ **Lemma 5.10** ($\star$)**.** *Reduction Rule 5.3 is α-safe.*

This leads to the following result.

▶ **Lemma 5.11** ($\star$)**.** *Suppose graph G is k-contractible to a cactus and none of the Reduction Rules 5.1, 5.2 and 5.3 are applicable on the instance (G, k). Then, $|V(G)|$ is $\mathcal{O}((2k)^{2d} + k^4)$.*

▶ **Theorem 5.12** ($\star$)**.** Cactus Contraction *admits a strict PSAKS with $\mathcal{O}((2k)^{2\lceil \frac{\alpha}{\alpha-1}+1\rceil} + k^5)$ vertices.*

References

1 T. Asano and T. Hirata. Edge-contraction problems. *Journal of Computer and System Sciences*, 26(2):197–208, 1983. doi:10.1016/0022-0000(83)90012-0.

2 R. Belmonte, P. A. Golovach, P. van 't Hof, and D. Paulusma. Parameterized complexity of three edge contraction problems with degree constraints. *Acta Informatica*, 51(7):473–497, 2014. doi:10.1007/s00236-014-0204-z.

3 H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039.

4 A. E. Brouwer and H. J. Veldman. Contractibility and NP-completeness. *Journal of Graph Theory*, 11(1):71–79, 1987.

5 L. Cai and C. Guo. *Contracting Few Edges to Remove Forbidden Induced Subgraphs*, pages 97–109. Springer International Publishing, 2013. doi:10.1007/978-3-319-03898-8_10.

6 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer-Verlag, 2015.

7 R. Diestel. *Graph Theory*. Springer-Verlag Berlin Heidelberg, 2000.

8 M. Dom, D. Lokshtanov, and S. Saurabh. Kernelization lower bounds through colors and IDs. *ACM Transactions on Algorithms (TALG)*, 11(2):13, 2014.

9 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer-Verlag London, 2013.

10 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

11 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H.Freeman and Company, 1979.

12 P. A. Golovach, M. Kamiński, D. Paulusma, and D. M. Thilikos. Increasing the minimum degree of a graph by contractions. *Theoretical Computer Science*, 481:74–84, 2013. doi:10.1016/j.tcs.2013.02.030.

13 P. A. Golovach, P. van 't Hof, and D. Paulusma. Obtaining planarity by contracting few edges. *Theoretical Computer Science*, 476:38–46, 2013. doi:10.1016/j.tcs.2012.12.041.

14 S. Guillemot and D. Marx. A faster FPT algorithm for Bipartite Contraction. *Information Processing Letters*, 113(22–24):906–912, 2013. doi:10.1016/j.ipl.2013.09.004.

15 P. Heggernes, P. van 't Hof, D. Lokshtanov, and C. Paul. Obtaining a bipartite graph by contracting few edges. *SIAM Journal on Discrete Mathematics*, 27(4):2143–2156, 2013. doi:10.1137/130907392.

16 P. Heggernes, P. van't Hof, B. Lévêque, D. Lokshtanov, and C. Paul. Contracting Graphs to Paths and Trees. *Algorithmica*, 68(1):109–132, 2014.

17 D. Lokshtanov, N. Misra, and S. Saurabh. *On the Hardness of Eliminating Small Induced Subgraphs by Contracting Edges*, pages 243–254. Springer International Publishing, 2013. doi:10.1007/978-3-319-03898-8_21.

18 D. Lokshtanov, F. Panolan, M. S. Ramanujan, and S. Saurabh. Lossy kernelization. *CoRR*, abs/1604.04111, 2016.

19 B. Martin and D. Paulusma. The computational complexity of disconnected cut and $2K_2$-partition. *Journal of Combinatorial Theory, Series B*, 111:17–37, 2015. doi:10.1016/j.jctb.2014.09.002.

20 T. Watanabe, T. Ae, and A. Nakamura. On the removal of forbidden graphs by edge-deletion or by edge-contraction. *Discrete Applied Mathematics*, 3(2):151–153, 1981. doi:10.1016/0166-218X(81)90039-1.

21 T. Watanabe, T. Ae, and A. Nakamura. On the NP-hardness of edge-deletion and -contraction problems. *Discrete Applied Mathematics*, 6(1):63–78, 1983. doi:10.1016/0166-218X(83)90101-4.