Contracting Graphs to Paths and Trees*

Pinar Heggernes[1], Pim van 't Hof[1], Benjamin Lévêque[2], Daniel Lokshtanov[3], and Christophe Paul[2]

[1] Department of Informatics, University of Bergen, N-5020 Bergen, Norway
{pinar.heggernes,pim.vanthof}@ii.uib.no
[2] CNRS, LIRMM, Université Montpellier 2, Montpellier, France
{leveque,paul}@lirmm.fr
[3] Dept. Computer Science and Engineering, University of California San Diego, USA
dlokshtanov@cs.ucsd.edu

Abstract. Vertex deletion and edge deletion problems play a central role in parameterized complexity. Examples include classical problems like Feedback Vertex Set, Odd Cycle Transversal, and Chordal Deletion. The study of analogous edge contraction problems has so far been left largely unexplored from a parameterized perspective. We consider two basic problems of this type: Tree Contraction and Path Contraction. These two problems take as input an undirected graph G on n vertices and an integer k, and the task is to determine whether we can obtain a tree or a path, respectively, by a sequence of at most k edge contractions in G. For Tree Contraction, we present a randomized $4^k\, n^{O(1)}$ time polynomial-space algorithm, as well as a deterministic $4.98^k\, n^{O(1)}$ time algorithm, based on a variant of the color coding technique of Alon, Yuster and Zwick. We also present a deterministic $2^{k+o(k)} + n^{O(1)}$ time algorithm for Path Contraction. Furthermore, we show that Path Contraction has a kernel with at most $5k + 3$ vertices, while Tree Contraction does not have a polynomial kernel unless NP $\subseteq$ coNP/poly. We find the latter result surprising because of the connection between Tree Contraction and Feedback Vertex Set, which is known to have a kernel with $4k^2$ vertices.

1 Introduction

For a graph class Π, the Π-Contraction problem takes as input a graph G on n vertices and an integer k, and the question is whether there is a graph $H \in \Pi$ such that G can be contracted to H by using at most k edge contractions. In early papers by Watanabe et al. [41, 42] and Asano and Hirata [3], Π-Contraction was proved to be NP-complete for several classes Π. The Π-Contraction problem fits into a wider and well studied family of graph modification problems, where vertex deletions and edge deletions are two other ways

* An extended abstract of this paper has been presented at the 6th International Symposium on Parameterized and Exact Computation (IPEC 2011) [28]. This work has been supported by the Research Council of Norway (project SCOPE, 197548/V30), the French ANR project AGAPE (ANR-09-BLAN-0159) and the Languedoc-Roussillon "Chercheur d'avenir" project KERNEL.

of modifying a graph. Π-Vertex Deletion and Π-Edge Deletion are the problems of deciding whether some graph belonging to graph class Π can be obtained from G by at most k vertex deletions or by at most k edge deletions, respectively. All of these problems are shown to be NP-complete for most of the interesting graph classes Π [37, 43–45]. However, whereas Π-Vertex Deletion and Π-Edge Deletion have been studied in detail for several graph classes Π with respect to fixed-parameter tractability (e.g., [4, 6, 11, 21, 27, 32, 34, 35, 38, 31]), this has not been the case for Π-Contraction. Note that every edge contraction reduces the number of vertices of the input graph by one, which means that the parameter k of Π-Contraction is never more than $n - 1$.

A *parameterized problem* is a subset $Q \subseteq \Sigma^* \times \mathbb{N}$ for some finite alphabet Σ, where the second part of the input is called the *parameter*. A parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is said to be *fixed-parameter tractable* (or FPT) if for each pair $(x, k) \in \Sigma^* \times \mathbb{N}$ it can be decided in time $f(k)\,|x|^{O(1)}$ whether $(x, k) \in Q$, for some function f that only depends on k; here, $|x|$ denotes the length of input x. We say that a parameterized problem Q has a *kernel* if there is an algorithm that transforms each instance (x, k) in time $(|x|+k)^{O(1)}$ into an instance (x', k'), such that $(x, k) \in Q$ if and only if $(x', k') \in Q$ and $|x'| + k' \leq g(k)$ for some function g. Here, g is typically an exponential function of k. If g is a polynomial or a linear function of k, then we say that the problem has a *polynomial* kernel or a *linear* kernel, respectively. We refer the interested reader to the monographs by Downey and Fellows [21] and Flum and Grohe [22] for more background on parameterized complexity.

It is known that a parameterized problem is fixed-parameter tractable if and only if it is decidable and has a kernel [21, 22], and several fixed-parameter tractable problems are known to have polynomial or even linear kernels. The recent establishment of methods for proving non-existence of polynomial kernels, under some complexity theoretical assumptions [7–9], significantly increased the interest in the question whether or not a fixed-parameter tractable problem has a polynomial kernel. During the last decade, considerable effort has also been devoted to improving the parameter dependence in the running time of classical parameterized problems. Even in the case of a running time $c^k\, n^{O(1)}$ for some constant c, lowering the base of the exponential function is considered to be an important challenge. For instance, the running time of Feedback Vertex Set has been successively improved from $37.7^k\, n^{O(1)}$ [26] to $10.57^k\, n^{O(1)}$ [18], $5^k\, n^{O(1)}$ [13], $3.83^k\, n^{O(1)}$ [12], and randomized $3^k\, n^{O(1)}$ [17].

In this paper, we present results along these established lines for Tree Contraction and Path Contraction. Since edge contractions preserve the number of connected components, we may assume that the input graph is connected. Hence Tree Contraction is equivalent to the Π-Contraction problem when Π is the class of acyclic graphs. We find these two problems of particular interest, since the corresponding vertex deletion problems Feedback Vertex Set and Longest Induced Path are famous and well studied. These two problems, when parameterized by the number of deleted vertices, are known to be fixed-parameter tractable and to have polynomial kernels [19, 21, 40]. NP-completeness

of Tree Contraction and Path Contraction readily follows from known results [3, 10]. It is easy to see that if a graph G is contractible to a path or a tree with at most k edge contractions, then the treewidth of G is at most $k+1$. Consequently, when parameterized by k, fixed-parameter tractability of Tree Contraction and Path Contraction follows from the well known result of Courcelle [15], as both problems are expressible in monadic second order logic. However, this approach yields very unpractical algorithms whose running times involve huge functions of k.

We give fast FPT algorithms for both problems, and study them from a kernelization point of view. For Tree Contraction, we present a polynomial-space randomized algorithm that runs in time $4^k\, n^{O(1)}$, as well as a deterministic $4.98^k\, n^{O(1)}$ time algorithm, based on a variant of the color coding technique of Alon, Yuster and Zwick [1]. We also show that Tree Contraction does not have a polynomial kernel, unless NP $\subseteq$ coNP/poly. This is in contrast with the corresponding vertex deletion problem Feedback Vertex Set, which is known to have a kernel with $4k^2$ vertices [40]. We then show that Path Contraction has a linear vertex kernel, and use this kernel to obtain a deterministic algorithm for the problem with running time $2^{k+o(k)} + n^{O(1)}$.2-

Let us finish this section by mentioning some related recent work. When restricted to chordal input graphs, Tree Contraction and Path Contraction can be solved in time $O(n+m)$ and $O(nm)$, respectively [29]. Very recently, Bipartite Contraction was shown to be fixed-parameter tractable [30]. Golovach et al. [25] studied the Degree Contraction problem, which takes as input a graph G and two integers d and k, and asks whether G can be modified into a graph of minimum degree at least d by contracting at most k edges in G. They proved that the problem is NP-complete already when $d = 14$. They also showed that the problem is fixed-parameter tractable when parameterized by k and d, but becomes W[1]-hard when parameterized by k. Finally, settling an open problem frequently posed in different settings, Martin and Paulusma [33] showed that Π-Contraction is NP-complete when Π is the class of bicliques $K_{p,q}$ with $p, q \geq 2$.

2 Definitions and Notation

All graphs in this paper are finite, undirected, and simple, i.e., do not contain multiple edges or loops. Given a graph G, we denote its vertex set by $V(G)$ and its edge set by $E(G)$. We also use the ordered pair $(V(G), E(G))$ to represent G. Throughout the paper, whenever we consider a problem or an algorithm, we use n and m to denote the number of vertices and edges, respectively, of the *input* graph.

Let $G = (V, E)$ be a graph. The *neighborhood* of a vertex v in G is the set $N_G(v) = \{w \in V \mid vw \in E\}$ of *neighbors* of v in G. Let $S \subseteq V$. We write $N_G(S)$ to denote $\bigcup_{v \in S} N_G(v) \setminus S$. We say that S *dominates* a set $T \subseteq V$ if every vertex in T either belongs to S or has at least one neighbor in S, and S is *independent* if no vertex in S has a neighbor in S. We write $G[S]$ to denote the subgraph of

G *induced* by S. We use shorthand notation $G - v$ to denote $G[V \setminus \{v\}]$ for a vertex $v \in V$, and $G - S$ to denote $G[V \setminus S]$ for a set of vertices $S \subseteq V$. Two disjoint sets $S_1, S_2 \subseteq V(G)$ are *adjacent* if there exists an edge $s_1s_2 \in E(G)$ with $s_1 \in S_1$ and $s_2 \in S_2$.

A graph is *connected* if it has a path between every pair of its vertices, and is *disconnected* otherwise. The *connected components* of a graph are its maximal connected subgraphs. We say that a vertex subset $S \subseteq V$ is *connected* if $G[S]$ is connected. A *bridge* in a connected graph is an edge whose deletion results in a disconnected graph. A *cut vertex* in a connected graph is a vertex whose deletion results in a disconnected graph. A graph is 2-*connected* if it has no cut vertex. A 2-*connected component* of a graph G is a maximal 2-connected subgraph of G. A *connected vertex cover* of a graph G is a subset $V' \subseteq V(G)$ such that $G[V']$ is connected and every edge of G has at least one endpoint in V'. We say that V' is a *minimum* connected vertex cover is there is no connected vertex cover V'' of G such that $|V''| < |V'|$.

We use P_ℓ to denote the graph isomorphic to a path on ℓ vertices, i.e., the graph with ordered vertex set $\{p_1, p_2, p_3, \ldots, p_\ell\}$ and edge set $\{p_1p_2, p_2p_3, \ldots, p_{\ell-1}p_\ell\}$. We will also write $p_1p_2 \cdots p_\ell$ to denote P_ℓ. A *tree* is a connected acyclic graph. A vertex of a tree with exactly one neighbor is called a *leaf*, and an *internal* vertex is a vertex that is not a leaf. A *star* is a tree isomorphic to the graph with vertex set $\{a, v_1, v_2, \ldots, v_s\}$ and edge set $\{av_1, av_2, \ldots, av_s\}$. Vertex a is called the *center* of the star. A graph $G = (V, E)$ is *bipartite* if there exist two disjoint sets $A, B \subseteq V$ such that $V = A \cup B$ and both A and B are independent sets; we use (A, B, E) to denote such a bipartite graph.

The *contraction* of the edge xy in G removes vertices x and y from G, and replaces them by a new vertex, which is made adjacent to precisely those vertices that were adjacent to at least one of the vertices x and y. A graph G is *contractible* to a graph H (or H-*contractible*) if H can be obtained from G by a sequence of edge contractions. Equivalently, G is contractible to H if there is a surjection $\varphi : V(G) \to V(H)$, with $W(h) = \{v \in V(G) \mid \varphi(v) = h\}$ for every $h \in V(H)$, that satisfies the following two conditions:

(i) $W(h)$ is a (non-empty) connected set in G, for every $h \in V(H)$;
(ii) $W(h_i)$ and $W(h_j)$ are adjacent in G if and only if $h_ih_j \in E(H)$, for every $h_i, h_j \in V(H)$.

We say that $\mathcal{W} = \{W(h) \mid h \in V(H)\}$ is an H-*witness structure* of G, and the sets $W(h)$, for $h \in V(H)$, are called *witness sets* of $\mathcal{W}$. Note that $\mathcal{W}$ is a partition of $V(G)$. For any subset $E' \subseteq E(G)$, we write G/E' to denote the graph obtained from G by contracting all the edges in E'. If E' consists of a single edge uv, then we write G/uv instead of $G/\{uv\}$. Similarly, for any subset $E' \subseteq E$ and any edge $uv \in E'$, we sometimes write $E' \setminus uv$ instead of $E' \setminus \{uv\}$.

If a witness set contains more than one vertex of G, then we call it a *big* witness set; a witness set consisting of a single vertex of G is called *small*. We say that G is k-*contractible* to H, with $k \leq n - 1$, if H can be obtained from G by at most k edge contractions, i.e., if there exists a set $E' \subseteq E(G)$ of at most

k edges such that G/E' is isomorphic to H. The next observation follows from the above definitions.

Observation 1 *If a graph G is k-contractible to a graph H, then $|V(G)| \leq |V(H)| + k$, and any H-witness structure $\mathcal{W}$ of G satisfies the following three properties:*

- *no witness set of $\mathcal{W}$ contains more than $k+1$ vertices;*
- *$\mathcal{W}$ has at most k big witness sets;*
- *all the big witness sets of $\mathcal{W}$ together contain at most $2k$ vertices.*

A 2-*coloring* of a graph G is a function $\phi : V(G) \to \{1, 2\}$. Here, a 2-coloring of G is merely an assignment of colors 1 and 2 to the vertices of G, and should not be confused with a *proper* 2-coloring of G, which is a 2-coloring with the additional property that no two adjacent vertices receive the same color. If all the vertices belonging to a set $S \subseteq V(G)$ have been assigned the same color by ϕ, we say that S is *monochromatic* with respect to ϕ, and we use $\phi(S)$ to denote the color of the vertices of S. Any 2-coloring ϕ of G defines a partition of $V(G)$ into two sets V_ϕ^1 and V_ϕ^2, which are the sets of vertices of G colored 1 and 2 by ϕ, respectively. A set $X \subseteq V(G)$ is a *monochromatic component* of ϕ if $G[X]$ is a connected component of $G[V_\phi^1]$ or a connected component of $G[V_\phi^2]$. We say that two different 2-colorings ϕ_1 and ϕ_2 of G *coincide* on a vertex set $A \subseteq V(G)$ if $\phi_1(v) = \phi_2(v)$ for every vertex $v \in A$.

Finally, we will use the following well-known result in the running time analysis of some of our algorithms (see for example [2], page 440).

Proposition 1. *For every $x \geq 1$,*

$$\left(1 - \frac{1}{x}\right)^x \leq 1/e \leq \left(1 - \frac{1}{x+1}\right)^x.$$

3 The Tree Contraction Problem

Asano and Hirata [3] showed that Tree Contraction is NP-complete. In this section, we first show that Tree Contraction does not have a polynomial kernel, unless NP $\subseteq$ coNP/poly. We then present a randomized algorithm for Tree Contraction that uses polynomial space and runs in time $4^k n^{O(1)}$, and show how it can be derandomized using families of perfect hash functions, yielding a deterministic $4.98^k n^{O(1)}$ time algorithm.

Recent breakthrough results by Bodlaender et al. [7] and Fortnow and Santhanam [23] provide a framework for proving that certain parameterized problems do not have polynomial kernels, unless NP $\subseteq$ coNP/poly. Bodlaender et al. [9] extended this framework by introducing the following concept. A *polynomial time and parameter transformation* (or *polynomial parameter transformation* for short) from a parameterized problem P to a parameterized problem Q is a polynomial-time function that transforms each instance (x, k) of P into an instance (x', k') of Q such that (x, k) is a yes-instance if and only if (x', k') is a

yes-instance, and $k' \leq p(k)$ for some polynomial p. For a parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$, the *derived classical problem* Q^c is the set $\{x\mathbf{1}^k \mid (x,k) \in Q\}$, where $\mathbf{1} \notin \Sigma$.

We use the following theorem, due to Bodlaender et al. [9], to prove our first result.

Theorem 1 ([9]). *Let P and Q be parameterized problems, and let P^c and Q^c be the derived classical problems. Suppose that P^c is NP-complete and Q^c is in NP. If there is a polynomial parameter transformation from P to Q and Q has a polynomial kernel, then P also has a polynomial kernel.*

Theorem 2. Tree Contraction *does not have a polynomial kernel, unless NP $\subseteq$ coNP/poly.*

Proof. We give a polynomial parameter transformation from Red-Blue Domination to Tree Contraction. The Red-Blue Domination problem takes as input a bipartite graph $G = (A, B, E)$ and an integer t, and the question is whether there exists a subset of at most t vertices in B that dominates A. We may assume that every vertex of A has a neighbor in B, and that $t \leq |A|$. This problem, when parameterized by $|A|$, has been shown not to have a polynomial kernel, unless NP $\subseteq$ coNP/poly [20]. By a straightforward reduction from the NP-complete problem Set Cover [24], the derived classical problem of Red-Blue Domination is NP-complete. Clearly, the derived classical problem of Tree Contraction is in NP. Hence, the existence of the polynomial parameter transformation described below, together with Theorem 1, implies that Tree Contraction does not have a kernel with size polynomial in k, unless NP $\subseteq$ coNP/poly.

Given an instance of Red-Blue Domination, i.e., a bipartite graph $G = (A, B, E)$ and an integer t, we construct an instance (G', k) of Tree Contraction with $G' = (A' \cup B', E')$ as follows. We set $k = |A| + t$. In order to construct G' from G, we first add a new vertex a to A and make it adjacent to every vertex of B. We define $A' = A \cup \{a\}$. We then add, for every vertex u of A, $k+1$ new vertices to B that are all made adjacent to exactly u and a. The set B' consists of the set B and the $|A|(k+1)$ newly added vertices. This completes the construction. We show that there is a subset of at most t vertices in B that dominates A in G if and only if G' is k-contractible to a tree.

Assume there exists a set $S \subseteq B$ of size at most t such that S dominates A in G. Vertex a is adjacent to all vertices of S, so the set $X = A \cup S \cup \{a\}$ is connected in G'. Note that all the vertices of G' that do not belong to X form an independent set in G. We can define a T-witness structure $\mathcal{W}$ of G', where T is a tree, as follows: let X be the only big witness set of $\mathcal{W}$, and let each vertex of $V(G') \setminus X$ form a singleton witness set of $\mathcal{W}$. Contracting all the edges of a spanning tree of $G[X]$ yields the tree T; in fact, T is a star. Since X has at most $|A| + t + 1 = k + 1$ vertices, any spanning tree of $G[X]$ has at most k edges. Hence G' is k-contractible to a tree.

For the reverse direction, assume that G' is k-contractible to a tree T, and let $\mathcal{W}$ be a T-witness structure of G'. Recall that every vertex of A has a neighbor

in B. Hence, for each $u \in A$, there are $k+1$ cycles that contain a, u, one vertex $b \in B$ and one vertex from $B' \setminus B$, such that these cycles pairwise intersect only in $\{a, u, b\}$. This implies that a and u must appear in the same witness set of $\mathcal{W}$, since otherwise at least $k+1$ edge contractions are needed in order to kill all the cycles containing both a and u. Consequently, there must be a witness set $W \in \mathcal{W}$ that contains all the vertices of $A \cup \{a\}$. Since all the vertices of $G' - W$ belong to B', they form an independent set in G'. This means that W is the only big witness set of $\mathcal{W}$, and T is in fact a star. Since G' is k-contractible to T, we know that $|W| \leq k+1$ by Observation 1. Suppose W contains a vertex $x \in B' \setminus B$. By construction, x is adjacent only to a and exactly one vertex $a' \in A$. Let b' be a neighbor of a' in B. Then we have $N_{G'}(x) \subseteq N_{G'}(b')$, so $W' = (W \setminus \{x\}) \cup \{b'\}$ is connected and $|W'| \leq |W|$. Hence, replacing W in $\mathcal{W}$ by W' yields a T'-witness structure for a tree T' on at least as many vertices as T. By repeating this argument, we may assume that W contains no vertices of $B' \setminus B$. Let $S = W \setminus A'$. Since W is a connected set of size at least 2 and A' is an independent set, we have that S dominates A'. Moreover, $|S| = |W| - |A| - 1 \leq k - |A| = t$. We conclude that S is a subset of at most t vertices in B that dominates A in G.

In order to see that the transformation described above is a polynomial parameter transformation, first observe that $k = t + |A| \leq 2|A|$. Moreover, in order to obtain G' from G, we added $|A|(k+1)+1 \leq 2|A|^2+|A|+1$ vertices, $|B|$ edges between a and the vertices of B, and two edges incident to each of the $|A|(k+1)$ vertices of $B' \setminus B$. Hence the size of the graph has increased by $O(|B|+|A|^2)$. □

As a contrast to this negative result, we will present two fast algorithms for Tree Contraction. The first one, a randomized algorithm with running time $4^k\, n^{O(1)}$, uses as a subroutine a fast randomized algorithm due to Cygan et al. [17] for finding connected vertex covers. We then derandomize this algorithm using the concept of *universal sets* and a fast deterministic algorithm for finding connected vertex covers due to Binkele-Raible and Fernau [5], resulting in a deterministic algorithm with a slightly worse running time $4.98^k\, n^{O(1)}$. The next lemma implies that we may assume the input graph of Tree Contraction to be 2-connected.

Lemma 1. *A connected graph is k-contractible to a tree if and only if each of its 2-connected components can be contracted to a tree, using at most k edge contractions in total.*

Proof. Let $G_1, \ldots, G_q$ be the 2-connected components of G. It is well-known that the 2-connected components partition the edges of G.

First suppose that for each 2-connected component G_i there is a subset of edges $E'_i \subseteq E(G_i)$ such that G_i/E'_i is a tree and $\sum_{i=1}^{q} |E'_i| \leq k$. Let $E' = \cup_{i=1}^{q} E'_i$ and consider the graph G/E'. Clearly $|E'| \leq k$, so it remains to prove that G/E' is a tree. For contradiction, suppose that G/E' contains a cycle C'. The edges in E' uniquely define a (G/E')-witness structure $\mathcal{W}$ of G, whose witness sets are exactly the vertex sets of the connected components of the graph $(V(G), E')$. Let us consider the witness sets of $\mathcal{W}$ in G that correspond to the vertices of C'

in G/E'. By definition, the witness set $W(c)$ is a connected set in G for each $c \in V(C')$, and for every $c_1, c_2 \in V(C')$, the witness sets $W(c_1)$ and $W(c_2)$ are adjacent in G if and only if c_1 and c_2 are adjacent in G/E'. Since C' is a cycle in G/E', it follows that there is a cycle C in G passing through the witness sets $W(c)$ for each $c \in V(C')$. Since $|E(C')| \geq 3$, C contains at least three edges e_1, e_2, e_3 of $E(C')$ such that the endpoints of e_i belong to different witness sets of $\mathcal{W}$, for $1 \leq i \leq 3$. By the construction of $\mathcal{W}$, this means that none of the edge e_1, e_2, e_3 belongs to E'. It is well-known that for any cycle in a graph, all the vertices and edges of that cycle belong to the same 2-connected component of the graph. This means that C is contained in a 2-connected component of G, say G_j. Since $\{e_1, e_2, e_3\} \cap E'_j = \emptyset$, the graph G_j/E'_j contains a cycle of length at least 3, contradicting the assumption that G_j/E'_j is a tree.

Now suppose that G is k-contractible to a tree, and let $E' \subseteq E(G)$ be a set of edges such that $|E'| \leq k$ and G/E' is a tree. Let F be a spanning forest of the graph $(V(G), E')$. Note that $|E(F)| \leq |E'| \leq k$, and that $G/E(F)$ is a tree. For $1 \leq i \leq q$, let $F_i = F[V(G_i)]$ be the subgraph of F that is contained in the 2-connected component G_i, and let $E'_i = E(F_i)$. We claim that G_i/E'_i is a tree for each $1 \leq i \leq q$. For any two vertices $u, v \in V(G_i)$, u and v are contained in the same tree of F_i if and only if they are contained in the same tree of F, as the 2-connected component G_i contains all the vertices of any path in G between u and v. Hence, for every tree T' in F_i there is a single tree T in F such that T' is a subtree of T. Moreover, for every edge $e \in E(G_i)$, G_i contains all the edges of all the cycles in G that contain e. This implies that G_i/E'_i is a tree, since a cycle in G_i/E'_i would imply a cycle in $G/E(F)$, contradicting the fact that $G/E(F)$ is a tree. We conclude that we can turn each 2-connected component G_i of G into a tree by contracting the edges in E'_i, using $\sum_{i=1}^{q} |E'_i| = |E(F)| \leq k$ edge contractions in total. □

The main idea for our algorithms for Tree Contraction is to use 2-colorings of the input graph G to find a T-witness structure for some tree T that G is k-contractible to (or conclude that such a witness structure does not exist). In order to make this more concrete, we introduce the following definition.

Definition 1. *Let G be a 2-connected graph, let T be a tree, and let $\mathcal{W}$ be a T-witness structure of G. A 2-coloring ϕ of G is* $\mathcal{W}$-compatible *if it satisfies the following two conditions:*

(1) every witness set of $\mathcal{W}$ is monochromatic with respect to ϕ;
(2) if $W(u)$ and $W(v)$ are big witness sets of $\mathcal{W}$ and uv is an edge of T, then $\phi(W(u)) \neq \phi(W(v))$.

Let G be a 2-connected graph that can be contracted to a tree T, and let $\mathcal{W}$ be a T-witness structure of G. Suppose we are given this graph G and a 2-coloring ϕ of G, but neither T nor $\mathcal{W}$ is given. In Lemma 2 below, we will show that if ϕ is $\mathcal{W}$-compatible, then we can use the monochromatic components of ϕ to compute a T'-witness structure $\mathcal{W}'$ of G, such that T' is some tree with at least as many vertices as T (possibly $T' = T$ and $\mathcal{W}' = \mathcal{W}$). We prove both a

deterministic and a randomized version of the lemma, as we need both versions in the algorithms for Tree Contraction presented later.

So how can we find a T'-witness structure $\mathcal{W}'$ as described above when we are given G and a $\mathcal{W}$-compatible 2-coloring ϕ of G, but do not know $\mathcal{W}$? Informally, we do this by finding a "star-like" partition of each monochromatic component X of ϕ, where one set of the partition is a minimum connected vertex cover of $G[X]$, and all the other sets have size 1. There are several fast parameterized algorithms for finding a minimum connected vertex cover of a graph. For our purposes, we will use both the currently fastest deterministic algorithm, due to Binkele-Raible and Fernau [5], and the fastest randomized algorithm, due to Cygan et al. [17]. The Connected Vertex Cover problem takes as input a graph G and an integer t, and asks whether or not G has a connected vertex cover of size at most t.

Proposition 2 ([5]). *There is a deterministic algorithm solving* Connected Vertex Cover *in time* $2.4882^t\, n^{O(1)}$*, using polynomial space.*

Proposition 3 ([17]). *There is a Monte Carlo algorithm solving* Connected Vertex Cover *in time* $2^t\, n^{O(1)}$*, using polynomial space. The algorithm cannot give false positives and may give false negatives with probability at most* $1/2$*.*

We point out that the algorithms in Propositions 2 and 3 can be used not only to decide whether or not G has a connected vertex cover of size at most t, but also to construct such a connected vertex cover (with probability at least $1/2$ in the case of the Monte Carlo algorithm) if one exists [5, 17]. We will need this in the proof of the following lemma, which plays a crucial role in our algorithms for Tree Contraction.

Lemma 2. *Let G be a 2-connected graph, and let ϕ be a 2-coloring of G. If G has a T-witness structure $\mathcal{W}$ for some tree T whose largest witness set has size d, and ϕ is $\mathcal{W}$-compatible, then we can compute a T'-witness structure of G for some tree T' with at least as many vertices as T*

(i) in time $2.4882^d\, n^{O(1)}$*, or*
(ii) in time $2^d\, n^{O(1)}$ *with probability at least* $1/e$*,*

using polynomial space.

Proof. We assume that G has at least 3 vertices; the lemma trivially holds otherwise. Suppose G has a T-witness structure $\mathcal{W}$ for some tree T such that the largest witness set of $\mathcal{W}$ has size d, and suppose that ϕ is $\mathcal{W}$-compatible. Let $\mathcal{X}$ be the set of monochromatic components of ϕ. We first make a few claims about $\mathcal{X}$. We refer to Figure 1 for a helpful illustration.

Claim 1. $\mathcal{X}$ is a T''-witness structure of G for some tree T'' that has at most as many vertices as T.

Proof Every witness set of $\mathcal{W}$ is monochromatic with respect to ϕ by property (1) in Definition 1, and connected by definition. Hence, for every $W \in \mathcal{W}$, there

exists an $X \in \mathcal{X}$ such that $W \subseteq X$. Moreover, since every $X \in \mathcal{X}$ is connected by definition, there exists a vertex subset $Y \subseteq V(T)$ such that $T[Y]$ is a connected subtree of T and $X = \bigcup_{y \in Y} W(y)$. Hence, $\mathcal{X}$ is a T''-witness structure of G for a tree T'' that has at most as many vertices as T. $\diamond$

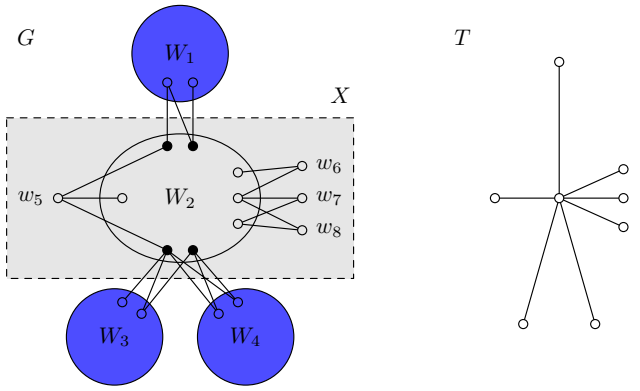

Fig. 1. Schematic illustration of a graph G (on the left) that is contractible to tree T (on the right). The sets W_1, W_2, W_3 and W_4 are connected, and G itself is 2-connected; for simplicity, not all vertices and edges have been drawn. Let $\mathcal{W}$ be the T-witness structure of G with four big witness sets $W_1, \ldots, W_4$ and four small witness sets $\{w_5\}, \ldots, \{w_8\}$. Let ϕ be a $\mathcal{W}$-compatible 2-coloring of G that colors the dark (blue) shaded big witness sets of $\mathcal{W}$ with one color, and the other sets of $\mathcal{W}$ with the other color. The set $\mathcal{X}$ of monochromatic components of ϕ forms a T''-witness structure of G, where T'' is a star with three leaves. Consider the monochromatic component X of φ that is indicated in the figure. The four witness sets of $\mathcal{W}$ that are contained in X form a shatter of X, but not necessarily a minimum shatter of X. The four black vertices form the set $\hat{X}$.

The idea behind our algorithm is to partition each $X \in \mathcal{X}$ into as many smaller witness sets as possible, such that we obtain either $\mathcal{W}$ or a T'-witness structure of G for some tree T' with at least as many vertices as T. In order to explain how we will replace $X \in \mathcal{X}$ by smaller sets, we need to introduce some additional terminology.

For any $X \in \mathcal{X}$, let $\hat{X} = N_G(V(G) \setminus X)$ be the set of vertices in X that have at least one neighbor outside X (the black vertices in Figure 1). A *shatter* of X is a partition of X into sets such that one of them is a connected vertex cover C of $G[X]$ containing every vertex of $\hat{X}$, and each of the others has size 1. The *size* of a shatter is the size of C, and a shatter of X is *minimum* if there is no shatter of X that has strictly smaller size, i.e., if C is a minimum-size superset of $\hat{X}$ that is a connected vertex cover of $G[X]$. Note that every set $X \in \mathcal{X}$ has a shatter, and that it is possible that $\{X\}$ is a shatter of X.

If follows from the proof of Claim 1 that for every $X \in \mathcal{X}$ there is a non-empty collection of witness sets $\mathcal{W}^* \subseteq \mathcal{W}$ such that X is the union of the sets in $\mathcal{W}^*$. The next claim shows that such a collection $\mathcal{W}^*$ forms a shatter of X.

Claim 2: Let $X \in \mathcal{X}$ and let $\mathcal{W}^* \subseteq \mathcal{W}$ be such that X is the union of the sets in $\mathcal{W}^*$. If $|X| \geq 2$, then

(i) exactly one of the sets in $\mathcal{W}^*$ is big; and
(ii) this big witness set is a connected vertex cover of $G[X]$ and contains $\hat{X}$.

Proof Suppose $|X| \geq 2$. Let $\mathcal{W}^* = \{W(v_1), \ldots, W(v_p)\}$ for some $p \geq 1$. The 2-connectedness of G implies that any small witness set in $\mathcal{W}$ (and in $\mathcal{W}^*$ in particular) corresponds to a leaf of T. Due to this observation and properties (1) and (2) in Definition 1, we know that at most one of the sets in $\mathcal{W}^*$ can be big. Suppose, for contradiction, that all the sets in $\mathcal{W}^*$ are small. Then $p \geq 2$ due to the assumption that $|X| \geq 2$. Since small witness sets correspond to leaves of T, each of the vertices $v_1, \ldots, v_p$ is a leaf in T. This implies that $p = 2$ and consequently $|V(G)| = 2$, contradicting the assumption that G has at least three vertices. Consequently, exactly one of the sets in $\mathcal{W}^*$, say $W(v_1)$, is big, and every set in $\mathcal{W}^* \setminus W(v_1)$ is small. Let $W(p_i) = \{w_i\}$ for $2 \leq i \leq p$. Since each of the sets $W(v_2), \ldots, W(v_p)$ is small, the vertices $v_2, \ldots, v_p$ are leaves in T and $\{w_2, \ldots, w_p\}$ is an independent set in G. This, together with the fact that witness sets are connected by definition, implies that $W(v_1)$ is a connected vertex cover of $G[X]$. Moreover, since $N_T(V(T) \setminus \{v_1, \ldots, v_p\}) = \{v_1\}$, the set $W(v_1)$ contains all the vertices of $\hat{X}$. $\diamond$

Claim 2 implies that if we replace every set $X \in \mathcal{X}$ with the sets of its shatter $\mathcal{W}^*$, consisting of exactly those sets of $\mathcal{W}$ that are contained in X, then we would obtain exactly the T-witness structure $\mathcal{W}$. However, recall that given G and ϕ, we only know $\mathcal{X}$, and not $\mathcal{W}$. The next claim shows that we can find a T'-witness structure of G for some tree T' with at least as many vertices as T, without having to use the sets of $\mathcal{W}$ in the process.

Claim 3: If we replace each set $X \in \mathcal{X}$ by the sets of a minimum shatter of X, then we obtain a T'-witness structure of G for some tree T' that has at least as many vertices as T.

Proof Let $X \in \mathcal{X}$. If $|X| = 1$, then $\mathcal{W}^* = \{X\}$ is the unique shatter of X. Suppose $|X| \geq 2$. As a result of Claim 2, X has a shatter $\mathcal{W}^*$ consisting of sets of $\mathcal{W}$; let $W(v_1)$ be the unique big witness set of $\mathcal{W}$ in $\mathcal{W}^*$, satisfying property (ii) of Claim 2. From the proof of Claim 2, it is clear that the sets $W(v_1), \ldots, W(v_p)$ of $\mathcal{W}^*$ define an S-witness structure $\mathcal{S}$ of the graph $G[X]$, where S is a star with $p - 1$ leaves.

We now consider a minimum shatter of X, and we let C denote the unique set in this minimum shatter that is a connected vertex cover of $G[X]$ containing all the vertices of $\hat{X}$. By the definition of a minimum shatter, there is no connected vertex cover C' of $G[X]$ such that C' contains all the vertices of $\hat{X}$ and $|C'| < |C|$. In particular, this means that $|C| \leq |W(v_1)|$. Moreover, since C is a vertex cover

of $G[X]$, the sets of the shatter define an S'-witness structure $\mathcal{S}'$ of the graph $G[X]$, where S' is a star with at least $p-1$ leaves. This means that replacing X in $\mathcal{X}$ by the sets of a minimum shatter of X is at least as good as replacing X by the sets of $\mathcal{W}'$. Since $\hat{X} \subseteq C$, we know that C is the only witness set of $\mathcal{S}'$ that is adjacent to witness sets of $\mathcal{X} \setminus X$, which ensures that the replacement of X by the sets of a minimum shatter still yields a witness structure for a tree. We conclude that if we replace each big witness set $X \in \mathcal{X}$ by the sets of a minimum shatter of X, then we obtain a T'-witness structure $\mathcal{W}'$ such that $|\mathcal{W}'| \geq |\mathcal{W}|$, where T' is a tree on at least as many vertices as T. $\diamond$

Due to Claim 3, it remains to prove that we can find a minimum shatter of each $X \in \mathcal{X}$ in the mentioned running time, with sufficient high success probability in the randomized case. We first describe a deterministic procedure, proving the first part of the lemma, before presenting the randomized version.

Proof of part (i) Let $X \in \mathcal{X}$. If $|X| = 1$, then a minimum shatter of X is simply the set X itself. Suppose $|X| \geq 2$. We then perform the following procedure to find a minimum shatter of X. Recall that $\hat{X} = N_G(V(G) \setminus X)$, and that we assumed the largest witness set of $\mathcal{W}$ to be of size d. We first construct a graph G' from the graph $G[X]$ by adding, for each vertex $x \in \hat{X}$, a new vertex x' and an edge xx'. Then we find a minimum connected vertex cover C of G' by applying the algorithm of Proposition 2 for all values of t from 1 up to d. Since ϕ is $\mathcal{W}$-compatible and each witness set of $\mathcal{W}$ has size at most d, such a set C will always be found. Observe that a minimum connected vertex cover of G' does not contain any vertex of degree 1, which implies that $\hat{X} \subseteq C$. From the definition of a minimum shatter and the minimality of C, it follows that C, together with the sets of size 1 formed by each of the vertices of $X \setminus C$, forms a minimum shatter of X. Due to Proposition 2, we can find a minimum shatter in $2.4882^d\, n^{O(1)}$ time for each set of $\mathcal{X}$. Since $\mathcal{X}$ contains at most n sets and all the other steps can be performed in polynomial time, the overall running time for the deterministic case is $2.4882^d\, n^{O(1)}$.

Proof of part (ii) In order to prove the second part of the lemma, we describe a randomized procedure that strongly resembles the above deterministic procedure. The only difference is that we do not use the algorithm of Proposition 2 to find a minimum connected vertex cover C of the graph G'. Instead, for each value of t from 1 to d, we run n times the algorithm of Proposition 3. Since each witness set of $\mathcal{W}$ has size at most d, G' has a minimum connected vertex cover of size c for some $c \leq d$. If we run the algorithm of Proposition 3 for $t = c$, we find a minimum connected vertex cover of G', and consequently a minimum shatter of X, with probability at least $1/2$. Repeating the algorithm n times for $t = c$ ensures that the probability of finding a minimum shatter of X is at least $1 - (1/2)^n$. We have to consider at most n sets in $\mathcal{X}$. The probability that we successfully find a minimum shatter of each of these sets is at least $(1-(1/2)^n)^n$. This means that the probability of finding the desired witness structure $\mathcal{W}'$ is at least

$$\left(1 - \frac{1}{2^n}\right)^n \geq \left(1 - \frac{1}{2^n}\right)^{2^n - 1} \geq 1/e\ ,$$

where the last inequality follows from Proposition 1. Note that we are only guaranteed to find $\mathcal{W}'$ in case we find a minimum shatter of each set in $\mathcal{X}$. For each $X \in \mathcal{X}$, we run the algorithm of Proposition 3 n times for each value of t from 1 to d, which takes $2^d\, n^{O(1)}$ time in total. Since $\mathcal{X}$ contains at most n sets and all the other steps can be performed in polynomial time, the overall running time is $2^d\, n^{O(1)}$ in the randomized case. □

We now present our randomized algorithm for Tree Contraction.

Theorem 3. *There is a Monte Carlo algorithm solving* Tree Contraction *in time* $4^k\, n^{O(1)}$*, using polynomial space. The algorithm cannot give false positives and may give false negatives with probability at most* $1/2$*.*

Proof. We describe such a randomized algorithm. Let (G, k) be an instance of Tree Contraction. Let us first assume that G is 2-connected. After presenting the algorithm, proving its correctness and analyzing its running time, we then turn our attention to the case where G is not 2-connected at the end of the proof.

The algorithm has an outer loop, which guesses the size d of the largest witness set of a possible T-witness structure $\mathcal{W}$ of G for a tree T. In particular, the outer loop iterates over all values of d from 1 to $k+1$. For each value of d, the algorithm runs the following inner loop, which is repeated 2^{2k-d+2} times. At each iteration of the inner loop, the algorithm generates a random 2-coloring ϕ of G by choosing, independently and uniformly at random, one of two possible colors for each vertex of G. It then runs the $2^d\, n^{O(1)}$ time randomized procedure described in the proof of Lemma 2 for computing a minimum shatter for each of the monochromatic components of ϕ, with the value d determined by the outer loop. If this procedure yields a T'-witness structure of G for a tree T' on at least $n-k$ vertices at some iteration of the inner loop, then the algorithm returns "yes". If none of the iterations of the inner loop yield such a witness structure, the outer loop picks the next value of d. If none of the iterations of the outer loop yield a T'-witness structure of G for a tree T' on at least $n-k$ vertices, then the algorithm returns "no".

Since the algorithm only outputs "yes" after it has found a tree T' such that G is k-contractible to T', the algorithm never gives false positives. Hence, in order to prove correctness, it remains to show that the algorithm answers "yes" with probability at least $1/2$ if G is k-contractible to a tree. Suppose G is k-contractible to a tree T, and let $\mathcal{W}$ be a T-witness structure of G whose largest witness set has size d^*. Note that $d^* \leq k+1$ by Observation 1. Let ψ be a 2-coloring of G such that each of the big witness sets of $\mathcal{W}$ is monochromatic with respect to ψ, such that $\psi(W(u)) \neq \psi(W(v))$ whenever both $W(u)$ and $W(v)$ are big witness sets and uv is an edge in T, and such that the vertices in the small witness sets are all colored 1. Observe that ψ is a $\mathcal{W}$-compatible 2-coloring of G, as is any other 2-coloring of G that coincides with ψ on all the vertices of the big witness sets of $\mathcal{W}$. The largest witness set of $\mathcal{W}$ requires d^*-1 edge contractions, after which our remaining budget of edge contractions is $k-(d^*-1) = k-d^*+1$. As a result of Observation 1, the total number of vertices contained in big witness sets is thus at most $d^* + 2(k-d^*+1) = 2k-d^*+2$. Hence, the probability that

a random 2-coloring ϕ coincides with ψ on all vertices contained in big witness sets, and thus is $\mathcal{W}$-compatible, is at least $1/(2^{2k-d^*+2})$.

Recall that our algorithm iterates over all values of d from 1 to $k+1$, and that $d^* \leq k+1$. At the correct iteration of the outer loop, i.e., at the iteration where $d = d^*$, a random 2-coloring of G is generated at each of the 2^{2k-d^*+2} iterations of the inner loop. The probability that none of these 2^{2k-d^*+2} random 2-colorings is $\mathcal{W}$-compatible is at most

$$\left(1 - \frac{1}{2^{2k-d^*+2}}\right)^{2^{2k-d^*+2}} \leq 1/e \ ,$$

where the inequality follows from Proposition 1. In other words, with probability at least $1/e$, the algorithm will generate a $\mathcal{W}$-compatible 2-coloring ϕ at some iteration of the inner loop when $d = d^*$ in the outer loop. When processing ϕ in this iteration, the algorithm will, as a result of Lemma 2, correctly conclude that G is k-contractible to a tree with probability at least $1/e$. We conclude that the algorithm correctly outputs "yes" with probability at least $1/e \cdot 1/e = 1/e^2$. Running the entire algorithm five times reduces the probability of false negatives to at most $(1 - \frac{1}{e^2})^5 < 1/2$.

Now let us analyze the running time of the algorithm. For each value of d, the inner loop is repeated 2^{2k-d+2} times, and each iteration of the inner loop takes $2^d\, n^{O(1)}$ time by Lemma 2. Hence, each iteration of the outer loop takes time $2^{2k-d+2}\, 2^d\, n^{O(1)} = 4^{k+1}\, n^{O(1)}$. Since the algorithm performs at most $k+1$ iterations of the outer loop, this yields an overall running time of $(k+1)4^{k+1}\, n^{O(1)} = 4^k\, n^{O(1)}$, where the equality follows from the observation that we may assume that $k \leq n$ and $n > 1$.

Finally, we consider the case where the input graph G is not 2-connected. A 2-connected component of G is *non-trivial* if it contains at least three vertices, and is *trivial* otherwise. Note that no edge contractions need to be made in trivial 2-connected components. Let $G_1, \ldots, G_q$ be the non-trivial 2-connected components of G. The algorithm outputs "no" if $q \geq k+1$. The correctness of this follows from the observation that in order to contract G to a tree, we need to contract at least one edge of every non-trivial 2-connected component of G. If $q = 0$, then G is already a tree and the algorithm outputs "yes". If $q = 1$, then we simply run the algorithm for the 2-connected case on the unique non-trivial 2-connected component of G. Now assume that $2 \leq q \leq k$.

For each non-trivial 2-connected component G_i and each value of k_i between 1 and k, we run the algorithm for the 2-connected case on the instance (G_i, k_i) $3 \log k$ times, and we let k_i^* be the smallest value of k_i for which at least one run of the algorithm on (G_i, k_i) outputs "yes". Since the algorithm for the 2-connected case does not give false positives, we know for sure that G_i is k_i^*-contractible to a tree. On the other hand, for a particular pair (G_i, k_i), if G_i is k_i-contractible to a tree, then the probability that no run of the algorithm on (G_i, k_i) outputs "yes" is at most $(\frac{1}{2})^{3 \log k} = \frac{1}{k^3}$. Since we have in total at most k^2 pairs (G_i, k_i), and by the union bound on probabilities, the probability that there is a pair (G_i, k_i) for which the algorithm fails to report that G_i is k_i-contractible to a

tree even though it is, is upper bounded by $k^2 \cdot \frac{1}{k^3} \leq 1/k$. If such a failure does not occur, then for every i we have that k_i^* is exactly the smallest value of k_i such that G_i is k_i-contractible to a tree. Finally, the algorithm answers "yes" if $\sum_{i=1}^{q} k_i^* \leq k$, and answers "no" otherwise. The correctness of this immediately follows from Lemma 1. Consequently, the algorithm cannot give false positives, and it may give false negatives with probability at most $1/k \leq 1/q \leq 1/2$, where the two inequalities follows from the assumption that $2 \leq q \leq k$.

Since we run the $4^k\, n^{O(1)}$ algorithm for the 2-connected case at most $k^2 \cdot 3 \log k$ times, the total time taken to determine all the k_i^*'s is $4^k\, n^{O(1)}$. All other steps of the algorithm take polynomial time. It is clear that the algorithm uses polynomial space. □

We now show that, at the cost of a slightly worse running time and exponential space usage, we can turn the algorithm of Theorem 3 into a deterministic algorithm for Tree Contraction. In order to derandomize our algorithm, we need to use the deterministic version of Lemma 2. We also need to avoid the use of random 2-colorings in our algorithm. This can be done by utilizing the notion of universal sets, defined as follows.

The *restriction* of a function $f : X \to Y$ to a set $S \subseteq X$ is the function $f_{|S} : S \to Y$ such that $f_{|S}(s) = f(s)$ for every $s \in S$. An (n,t)-*universal set* $\mathcal{F}$ is a set of functions from $\{1, 2, \ldots, n\}$ to $\{1, 2\}$ such that for every $S \subseteq \{1, 2, \ldots, n\}$ with $|S| = t$ the set $\mathcal{F}_{|S} = \{f_{|S} \mid f \in \mathcal{F}\}$ is equal to the set 2^S of all the functions from S to $\{1, 2\}$. It is clear from the definition that an (n,t)-universal set can be interpreted as a set of 2-colorings of a graph on n vertices, and this is what we will do in the proof of Theorem 5 below. The notion of (n,t)-universal sets was introduced by Naor, Schulman and Srinivasan [36]. The following result is due to Chen et al. [14], and is based on the construction of (n,t)-universal sets proposed in [36].

Theorem 4 ([14]). *There is an $O(n2^{t+12\log^2 t})$ time deterministic algorithm that constructs an (n,t)-universal set of size bounded by $n2^{t+12\log^2 t+2}$.*

Theorem 5. Tree Contraction *can be solved in time $4.98^k\, n^{O(1)}$.*

Proof. Let us first assume that the input graph G is 2-connected. We describe a deterministic algorithm for Tree Contraction that strongly resembles the randomized algorithm of Theorem 3. The only three differences between the two algorithms are as follows. Firstly, for each value of d, the deterministic algorithm constructs an $(n, 2k-d+2)$-universal set $\mathcal{F}_d$ using the algorithm of Theorem 4. Secondly, the inner loop of the deterministic algorithm iterates over all 2-colorings $\phi \in \mathcal{F}_d$ instead of generating a random 2-coloring at every iteration. Thirdly, in each iteration of the inner loop, the deterministic algorithm uses the $2.4882^d\, n^{O(1)}$ time deterministic procedure described in the proof of Lemma 2 instead of the $2^d\, n^{O(1)}$ time randomized one.

To prove correctness of the deterministic algorithm, it suffices to show that the algorithm always outputs "yes" if G is k-contractible to a tree. Suppose G is k-contractible to a tree T. Let $\mathcal{W}$ be a T-witness structure of G, and let

d^* be the size of the largest witness set of $\mathcal{W}$. As we argued in the proof of Theorem 3, the total number of vertices contained in big witness sets of $\mathcal{W}$ is at most $2k - d^* + 2$. At the iteration of the outer loop where $d = d^*$, the algorithm generates an $(n, 2k-d^*+2)$-universal set $\mathcal{F}_{d^*}$, after which the inner loop iterates over all 2-colorings in $\mathcal{F}_{d^*}$. By the definition of a universal set, the vertices in the big witness sets are 2-colored in all possible ways by the colorings in $\mathcal{F}_{d^*}$. Hence, $\mathcal{F}_{d^*}$ contains at least one $\mathcal{W}$-compatible 2-coloring ϕ. When processing ϕ, the algorithm will correctly output "yes" as a result of Lemma 2.

Let us analyze the running time of this deterministic algorithm. For each d, the size of $\mathcal{F}_d$ is at most $n2^{2k-d+2+12\log^2(2k-d+2)+2}$, and $\mathcal{F}_d$ can be constructed in $O(n2^{2k-d+2+12\log^2(2k-d+2)})$ time, by Theorem 4. The inner loop iterates over all 2-colorings in $\mathcal{F}_d$, and each iteration takes $2.4882^d\, n^{O(1)}$ time by Lemma 2. Since the outer loop iterates over all integer values of d from 1 to $k+1$, the overall running time of the algorithm is $\sum_{d=1}^{k+1} |\mathcal{F}_d| \cdot 2.4882^d\, n^{O(1)}$. Note that

$$\begin{aligned}\textstyle\sum_{i=1}^{k+1} 2^{2k-d} \cdot 2.4882^d &= 2^{2k} \cdot \textstyle\sum_{i=1}^{k+1} 1.2441^d \\ &\leq 2^{2k} \cdot (k+1) \cdot 1.2441^{k+1} \\ &= 1.2441 \cdot (k+1) \cdot 4.9764^k\ .\end{aligned}$$

We conclude that the overall running time of the described deterministic algorithm is $4.98^k\, n^{O(1)}$. Here, we slightly rounded up the base of the exponent in order to hide all other factors in the running time, including the factor $2^{12\log^2(2k-d+2)}$.

Note that if the input graph G is not 2-connected, we can easily adapt the deterministic algorithm in the way described in the last two paragraphs of the proof of Theorem 3, without increasing the running time, such that it considers the non-trivial 2-connected components of G one by one. □

4 The Path Contraction Problem

Brouwer and Veldman [10] showed that, for every fixed $\ell \geq 4$, it is NP-complete to decide whether a graph can be contracted to the path P_ℓ. This, together with the observation that a graph G is k-contractible to a path if and only if G is contractible to P_{n-k}, implies that Path Contraction is NP-complete. In the observation below, we describe a simple brute-force procedure for solving Path Contraction, which will be used as a subroutine in the algorithm presented in Theorem 7. Throughout this section, whenever we mention a P_ℓ-witness structure $\mathcal{W} = \{W_1, \ldots W_\ell\}$, it will be implicit that $P_\ell = p_1 \cdots p_\ell$, and $W_i = W(p_i)$ for every $i \in \{1, \ldots, \ell\}$.

Observation 2 *For any graph G and integer ℓ, there is a $2^n\, n^{O(1)}$ time algorithm that decides whether G is P_ℓ-contractible, and if so, generates all P_ℓ-witness structures of G.*

Proof. Let G be a graph that is P_ℓ-contractible for some integer ℓ. Let $\mathcal{W} = \{W_1, \ldots, W_\ell\}$ be a P_ℓ-witness structure of G. Consider a 2-coloring ϕ of G

that colors the set W_i with color 1 if i is odd and with color 2 otherwise, for $i = 1, \ldots, \ell$. Note that the witness sets in $\mathcal{W}$ are exactly the monochromatic components of ϕ. In other words, an edge uv of G is monochromatic if and only if both u and v belong to the same witness set W_i. Hence, if we repeatedly contract monochromatic edges, we end up with the path P_ℓ. Now consider the algorithm that generates all 2^n 2-colorings of G, and tests for each 2-coloring in $n^{O(1)}$ time whether contracting all monochromatic edges yields the path P_ℓ. It is clear that this algorithm decides in $2^n\, n^{O(1)}$ time whether or not G is P_ℓ-contractible, and that it generates all P_ℓ-witness structures of G if such witness structures exist. □

In the remainder of this section, we first show that Path Contraction has a linear vertex kernel. We then present a deterministic algorithm with running time $2^{k+o(k)} + n^{O(1)}$ for this problem. Consider the following reduction rule (see also Figure 2 for an illustration).

Rule 1 *Let (G, k) be an instance of* Path Contraction. *If G contains a bridge uv such that the deletion of edge uv from G results in two connected components that contain at least $k+2$ vertices each, then transform the instance into (G', k), where G' is the graph resulting from the contraction of edge uv.*

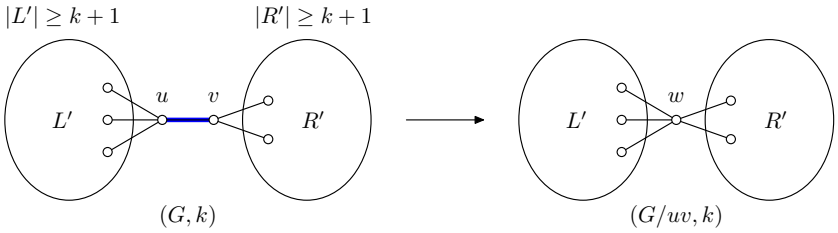

Fig. 2. An illustration of Rule 1: bridge uv is contracted, since deleting uv from G results in two connected components that contain at least $k + 2$ vertices each. Vertex w in the graph G/uv is the vertex resulting from the contraction of the edge uv in G.

The following lemma shows that the above reduction rule is "safe", i.e., that it returns an equivalent instance of the problem. We then prove that this single reduction rule yields a linear vertex kernel for Path Contraction.

Lemma 3. *Let (G', k) be an instance of* Path Contraction *resulting from the application of Rule 1 on (G, k). Then G' is k-contractible to a path if and only if G is k-contractible to a path.*

Proof. Let (G, k) be an instance of Path Contraction on which Rule 1 is applicable, and let uv be the bridge of G that is contracted to obtain G'. Let G_1 and G_2 be the two connected components that we obtain if we delete the

edge uv from G, with $L = V(G_1)$ and $R = V(G_2)$, such that $u \in L$ and $v \in R$. Furthermore, let $L' = L \setminus \{u\}$ and $R' = R \setminus \{v\}$, and let w be the vertex of G' resulting from the contraction of uv in G (see Figure 2).

Assume that G is k-contractible to a path, and let $E' \subseteq E(G)$ with $|E'| \leq k$ be a set of edges such that G/E' is a path. Suppose $uv \in E'$. Then $G'/(E' \setminus uv)$ is a path, since $G'/(E' \setminus uv) = (G/uv)/(E' \setminus uv) = G/E'$. Hence G' is $(k-1)$-contractible to a path. Now suppose $uv \notin E'$. Observe that $G'/E' = (G/uv)/E' = (G/E')/uv$. Since G/E' is a path, it is clear that $(G/E')/uv$, and consequently G'/E', is also a path. Hence G' is k-contractible to a path.

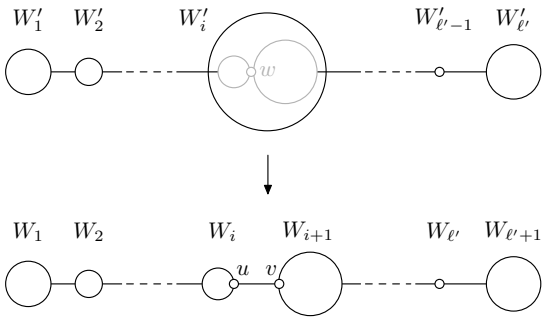

Fig. 3. A $P_{\ell'}$-witness structure $\mathcal{W}'$ of G' (top), and a $P_{\ell'+1}$-witness structure $\mathcal{W}$ of G (bottom) obtained from "splitting" the witness set W'_i into two new witness sets W_i and W_{i+1}, keeping all other witness sets the same.

For the other direction, assume that G' is k-contractible to a path $P_{\ell'}$, and let $\mathcal{W}' = \{W'_1, \ldots, W'_{\ell'}\}$ be a $P_{\ell'}$-witness structure of G' (see the top half of Figure 3). Let W'_i be the witness set of $\mathcal{W}'$ containing w. Since G_1 and G_2 contain at least two vertices each, w is a cut vertex of G'. The set $A = \bigcup_{j=1}^{i-1} W'_j$ is connected in $G' - w = G - \{u, v\}$. Hence A cannot contain vertices from both L' and R', so it contains only elements of L' or only elements of R'. Similarly, the set $B = \bigcup_{j=i+1}^{\ell} W_j$ contains only elements of L', or only elements of R'. By Observation 1, the set W'_i has size at most $k+1$. Hence W'_i, which contains w, does not contain all of L', as $|L' \cup \{w\}| \geq k+2$. Similarly, W'_i does not contain all of R', as $|R' \cup \{w\}| \geq k+2$. Hence, neither A nor B is empty, one contains only elements of R' and the other only elements of L'. Consequently, by replacing W'_i by two new sets $(W'_i \cap L') \cup \{u\}$ and $(W'_i \cap R') \cup \{v\}$, and keeping all other witness sets of $\mathcal{W}'$ the same, we obtain a $P_{\ell'+1}$-witness structure $\mathcal{W}$ of G (see Figure 3). Hence G is k-contractible to a path. □

We say that an instance (G, k) of Path Contraction is *reduced* if Rule 1 cannot be applied on (G, k).

Lemma 4. *Let (G, k) be a reduced instance of* Path Contraction. *If (G, k) is a yes-instance, then G has at most $5k + 3$ vertices.*

Proof. Suppose (G, k) is a reduced instance of Path Contraction, and suppose that (G, k) is a yes-instance. Then G is k-contractible to a path P_ℓ for some $\ell \geq n - k$. Let $\mathcal{W} = \{W_1, \ldots, W_\ell\}$ be a P_ℓ-witness structure of G. First assume that $\ell \leq 2k + 3$. Then, since $\ell \geq n - k$, we have $n \leq 3k + 3$. Now assume that $\ell \geq 2k + 4$, and let i be such that $k + 2 \leq i \leq \ell - k - 2$. Suppose, for contradiction, that both W_i and W_{i+1} are small witness sets, i.e., $W_i = \{u\}$ and $W_{i+1} = \{v\}$ for two vertices u and v of G (see the top half of Figure 4). Then uv forms a bridge in G whose deletion results in two connected components. Each of these components contains at least all vertices from $W_1, \ldots, W_{k+2}$ or all vertices from $W_{\ell-k-1}, \ldots, W_\ell$. Hence they contain at least $k + 2$ vertices each. Consequently, Rule 1 can be applied, contradicting the assumption that (G, k) is a reduced instance. So there are no two consecutive small witness sets among $W_{k+2}, \ldots, W_{\ell-k-1}$ (see the bottom half of Figure 4). Since $\mathcal{W}$ contains at most k big witness sets by Observation 1, at most $k + 1$ small witness sets can appear among $W_{k+2}, \ldots, W_{\ell-k-1}$. Hence $(\ell - k - 1) - (k + 2) + 1 \leq 2k + 1$, implying $\ell \leq 4k + 3$. Combining this with the earlier assumption that $\ell \geq n - k$ yields $n \leq 5k + 3$. □

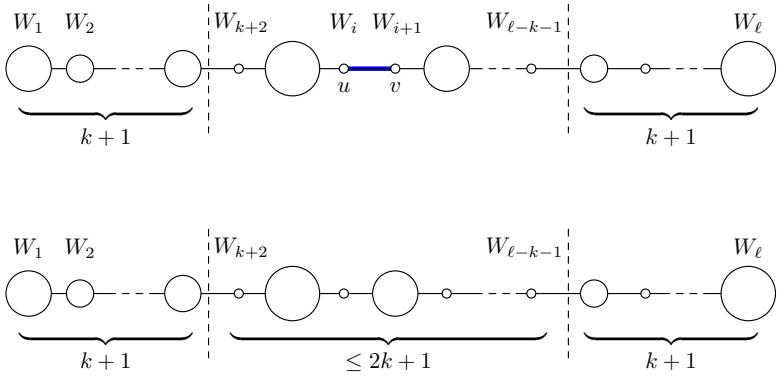

Fig. 4. Illustration for the proof of Lemma 4. If among the witness sets $W_{k+2}, \ldots, W_{\ell-k-1}$ two consecutive witness sets W_i and W_{i+1} are small (top), then Rule 1 can be applied on the instance (G, k). If no two consecutive witness sets among $W_{k+2}, \ldots, W_{\ell-k-1}$ are small (bottom), then $\ell \leq 4k + 3$.

Theorem 6. Path Contraction *has a kernel with at most* $5k + 3$ *vertices.*

Proof. We describe a polynomial-time kernelization algorithm for Path Contraction. Given an instance (G, k), the algorithm repeatedly tests, in linear time, whether Rule 1 can be applied on the instance under consideration, and applies the reduction rule if possible. Each application of Rule 1 strictly decreases the number of vertices. Hence, starting from the instance (G, k), we reach in

polynomial time a reduced instance (G', k) on which Rule 1 cannot be applied anymore. The algorithm outputs the instance (G', k) if $|V(G')| \leq 5k + 3$, and outputs a trivial no-instance of constant size otherwise. By Lemma 3, we know that G' is k-contractible to a path if and only if G is k-contractible to a path. Moreover, G' is not k-contractible to a path if $|V(G')| > 5k + 3$ as a result of Lemma 4. This proves the correctness of the described kernelization algorithm. □

Theorem 6 easily implies a $32^k\, k^{O(1)} + n^{O(1)}$ time algorithm for Path Contraction: given an instance (G, k) of the problem, transform it into an equivalent instance (G', k) in time $n^{O(1)}$, such that G' has at most $5k+3$ vertices, and then use Observation 2 to decide whether G' is k-contractible to a path. The natural follow-up question, which we answer affirmatively below, is whether this running time can be significantly improved.

Theorem 7. Path Contraction *can be solved in time* $2^{k+o(k)} + n^{O(1)}$.

Proof. Given an instance (G, k) of Path Contraction, our algorithm first constructs an equivalent instance (G', k) such that G' has at most $5k+3$ vertices. This can be done in $n^{O(1)}$ time by Theorem 6. For the rest of the proof, we assume that the input graph G has $n \leq 5k + 3$ vertices. The algorithm distinguishes between the following two cases.

Case 1: G has at most $k + \lfloor\sqrt{k}\rfloor$ vertices.

In this case, the algorithm runs the procedure described in the proof of Observation 2 to determine, for all values of ℓ from n down to 1, whether or not G is P_ℓ-contractible. This way, it finds the largest integer ℓ' for which G is $P_{\ell'}$-contractible. The algorithm outputs "yes" if $\ell' \geq n - k$, and outputs "no" otherwise.

Case 2: G has more than $k + \lfloor\sqrt{k}\rfloor$ vertices.

In this case, the algorithm first generates all subsets $W \subseteq V(G)$ of size at most $(5k+3)/\lfloor\sqrt{k}\rfloor$. Then, for each generated subset W, it checks whether W satisfies the following three conditions:

- the graph $G - W$ has at most $(5k + 3)/\lfloor\sqrt{k}\rfloor + 1$ connected components;
- all the connected components of $G - W$ have at most $k + \lfloor\sqrt{k}\rfloor - 1$ vertices;
- contracting each connected component of $G - W$ into a single vertex and contracting each connected component of $G[W]$ into a single vertex yields a path.

If a set W does not satisfy all three conditions, then it is discarded. Each set W that is not discarded, is called a *candidate*. For each candidate W, the algorithm considers each connected component G_i of $G - W$. Note that G_i is adjacent to one or two connected components of $G[W]$ by the definition of a candidate. The algorithm distinguishes between the two different cases as follows.

If $V(G_i)$ is adjacent to exactly two connected components L and R of $G[W]$, then the algorithm checks whether G_i is $P_{\lfloor\sqrt{k}\rfloor-1}$-contractible and if so, generates all $P_{\lfloor\sqrt{k}\rfloor-1}$-witness structures of G_i using the procedure described in Observation 2. For each such witness structure $\mathcal{X} = \{X_1, \ldots, X_{\lfloor\sqrt{k}\rfloor-1}\}$, it checks whether X_1 is the only set of $\mathcal{X}$ that is adjacent to L and $X_{\lfloor\sqrt{k}\rfloor-1}$ is the only set of $\mathcal{X}$ that is adjacent to R, or vice versa. If so, we let $\mathcal{X}_i$ denote this $P_{\lfloor\sqrt{k}\rfloor-1}$-witness structure of G_i, and the algorithm proceeds to the next connected component G_i. Otherwise, the algorithm discards W and proceeds to the next set W, or outputs "no" if all sets W have been processed.

If $V(G_i)$ is adjacent to exactly one connected component L of $G[W]$, then the algorithm does as follows. For all integer values of ℓ from $\lfloor\sqrt{k}\rfloor - 1$ down to 1, the algorithms checks whether G_i is P_ℓ-contractible and if so, generates all P_ℓ-witness structures of G_i using the procedure described in Observation 2. For each P_ℓ-witness structure $\mathcal{X} = \{X_1, \ldots, X_\ell\}$ of G_i, it checks whether either X_1 or X_ℓ is the only set of $\mathcal{X}$ that is adjacent to L. Let ℓ' be the largest value of ℓ for which G_i has such a witness structure, and let $\mathcal{X}_i$ denote the corresponding $P_{\ell'}$-witness structure of G_i. Note that $\ell' \geq 1$, since G_i clearly has a trivial P_1-witness structure (with $V(G_i)$ as its single witness set) that satisfies the required property.

Suppose the algorithm successfully processed all connected components of $G - W$, and obtained a witness structure $\mathcal{X}_i$ for each connected component G_i. The algorithm now checks whether all these witness structures $\mathcal{X}_i$, together with the vertex sets of the connected components of $G[W]$, form a P-witness structure of G for some path P on at least $n - k$ vertices. Note that this can easily be done in polynomial time by contracting all edges that have both endpoints in the same witness set, or in the same connected component of $G[W]$, and checking whether the obtained graph is a path on at least $n - k$ vertices. The algorithm outputs "yes" if this is the case. Otherwise, the algorithm tries the next subset W, or outputs "no" if all subsets W have been considered.

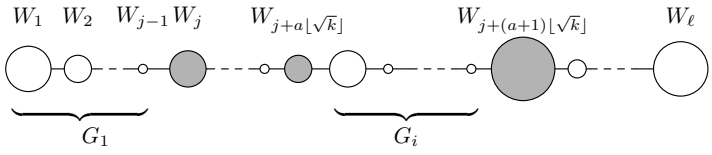

Fig. 5. Schematic illustration of the P_ℓ-witness structure $\mathcal{W}$ of G, mentioned in the correctness proof of the algorithm in Theorem 7. The shaded witness sets form the set W_j^*. The vertices in the sets $W_1, \ldots, W_{j-1}$ induce the graph G_1, which is an end-component of $G - W_j^*$. Let G_i be a connected component of $G - W_j^*$ that is not an end-component, i.e., G_i is the graph induced by the vertices of those witness sets of $\mathcal{W}$ that appear between the sets $W_{j+a\lfloor\sqrt{k}\rfloor}$ and $W_{j+(a+1)\lfloor\sqrt{k}\rfloor}$ for some integer a. Note that these witness sets $W_{j+a\lfloor\sqrt{k}\rfloor+1}, \ldots, W_{j+(a+1)\lfloor\sqrt{k}\rfloor-1}$ constitute a $P_{\lfloor\sqrt{k}\rfloor-1}$-witness structure $\mathcal{X}_i$ of G_i.

We now prove the correctness of the algorithm described above. The correctness of Case 1 follows from the observation that G is k-contractible to a path if and only if G is P_ℓ-contractible for some $\ell \geq n - k$.

Let us now analyze Case 2. Suppose G is k-contractible to a path P_ℓ for some $\ell \geq n - k$. Since G has more than $k + \lfloor\sqrt{k}\rfloor$ vertices in Case 2, we know that $\ell > \lfloor\sqrt{k}\rfloor$. Let $\mathcal{W} = \{W_1, \ldots, W_\ell\}$ be a P_ℓ-witness structure of G; see Figure 5 for an illustration. We define $W_i^* = W_i \cup W_{i+\lfloor\sqrt{k}\rfloor} \cup W_{i+2\lfloor\sqrt{k}\rfloor} \cup \ldots$, for each integer i with $1 \leq i \leq \lfloor\sqrt{k}\rfloor$. Since G has at most $5k + 3$ vertices, there is at least one index j such that $|W_j^*| \leq (5k + 3)/\lfloor\sqrt{k}\rfloor$. Let $G_1, \ldots, G_p$ denote the connected components of $G - W_j^*$, where $p \leq (5k+3)/\lfloor\sqrt{k}\rfloor + 1$. A connected component of $G - W_j^*$ that is adjacent to exactly one of the connected components of $G[W_j^*]$ is called an *end-component*. Each connected component G_i has a P'-witness structure $\mathcal{X}_i$ for some path P' on at most $\lfloor\sqrt{k}\rfloor - 1$ vertices, such that the sets of $\mathcal{X}_i$ are exactly those sets of $\mathcal{W}$ that contain the vertices of G_i. Note that the path P' has exactly $\lfloor\sqrt{k}\rfloor - 1$ vertices if G_i is not an end-component (see Figure 5). Moreover, the union of all these witness structures $\mathcal{X}_i$, together with the sets $W_j, W_{j+\lfloor\sqrt{k}\rfloor}, \ldots$, clearly forms the P_ℓ-witness structure $\mathcal{W}$ of G. Each connected component G_i has at most $k + \lfloor\sqrt{k}\rfloor - 1$ vertices by Observation 1. In fact, it is easy to see that W_j^* satisfies each of the three properties of a candidate, defined in Case 2 of the algorithm. Hence, at some point, the set W that the algorithm considers will be exactly the set W_j^*. From the above it is clear that in this case, the algorithm will correctly output "yes".

It remains to perform a running time analysis. Since G has $k + \sqrt{k}$ vertices in Case 1, the algorithm runs the procedure of Observation 2 at most $k + \sqrt{k}$ times, each time taking $2^{k+\sqrt{k}}(k + \sqrt{k})^{O(1)} = 2^{k+o(k)}$ time. Hence the overall running time in Case 1 is $2^{k+o(k)}$. In Case 2, the algorithm considers no more than $(5k + 3)^{(5k+3)/\lfloor\sqrt{k}\rfloor} = 2^{o(k)}$ subsets $W \subseteq V(G)$. For each generated set W, the $2^{k+o(k)}$ time procedure of Observation 2 is performed at most $\lfloor\sqrt{k}\rfloor - 1$ times on each of the $O(\sqrt{k})$ connected components of $G - W$. Since all additional checks can clearly be performed in polynomial time and G has at most $5k + 3$ vertices, we get a total running time of $2^{k+o(k)}$ also for this case. □

5 Concluding Remarks

The number of edges to contract in order to obtain a certain graph property is a natural measure of how close the input graph is to having that property, similar to the more established similarity measures of the number of edges or vertices to delete. The latter measures are well studied when the desired property is being acyclic or being a path, defining some of the most widely known and well studied problems within parameterized complexity. Inspired by this, we gave kernelization results and fast fixed-parameter algorithms for Tree Contraction and Path Contraction. We think our results motivate the parameterized study of similar problems, an example of which is Interval Contraction. It is not known whether the vertex deletion variant of this problem, Interval

Vertex Deletion, is fixed-parameter tractable. Is Interval Contraction fixed-parameter tractable?

As is common in the initial study of the parameterized complexity of graph modification problems, we chose the number of allowed operations k to be the parameter in our study of Tree Contraction and Path Contraction. An interesting question is whether similar positive results for these two problems can be obtained with respect to other parameters, such as the treewidth t of the input graph. Courcelle's Theorem implies that both problems are fixed-parameter tractable with this parameter, and it is possible to solve both problems in time $t^{O(t)}\, n^{O(1)}$ using a dynamic programming approach. However, a much more challenging task is to decide whether the problems can be solved in time $c^t\, n^{O(1)}$ for some constant c. Recently, Cygan et al. [17] developed a powerful technique for identifying connectivity problems that allow (randomized) algorithms with this time complexity. Soon after, Pilipczuk [39] provided a logical framework for obtaining similar results. We point out that it does not seem easy at all to apply the Cut&Count technique of Cygan et al. [17] to our problems, nor is it clear how to formulate our problems in Pilipczuk's [39] existential counting modal logic. Hence, a further study of the problems Tree Contraction and Path Contraction with respect to different parameters such as treewidth seems to be an interesting direction for further research.

Very recently, Cygan [16] announced a deterministic algorithm solving Connected Vertex Cover in time $2^t\, n^{O(1)}$, using polynomial space. If we use Cygan's algorithm instead of the algorithms of Propositions 2 and 3, adapting Lemma 2 accordingly, then the running time of our deterministic algorithm for Tree Contraction drops to $4^k\, n^{O(1)}$.

Acknowledgments. The authors would like to thank Jesper Nederlof, Saket Saurabh, Erik Jan van Leeuwen and Martin Vatshelle for valuable suggestions and comments. We are also indebted to the three anonymous referees, whose detailed comments and suggestions helped us to correct small mistakes, simplify proofs and significantly improve the overall presentation of the paper.

References

1. Alon, N., Yuster, R., Zwick, U.: Color-coding. Journal of the ACM 42(4), 844–856 (1995)
2. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press, New York, NY, USA, 1st edn. (2009)
3. Asano, T., Hirata, T.: Edge-contraction problems. Journal of Computer and System Sciences 26(2), 197–208 (1983)
4. van Bevern, R., Komusiewicz, C., Moser, H., Niedermeier, R.: Measuring indifference: Unit interval vertex deletion. In: Thilikos, D.M. (ed.) 36th International Workshop on Graph Theoretic Concepts in Computer Science, WG 2010. Lecture Notes in Computer Science, vol. 6410, pp. 232–243. Springer (2010)
5. Binkele-Raible, D., Fernau, H.: Enumerate and measure: Improving parameter budget management. In: Raman, V., Saurabh, S. (eds.) 5th International Symposium

on Parameterized and Exact Computation, IPEC 2010. Lecture Notes in Computer Science, vol. 6478, pp. 38–49. Springer (2010)
6. Bodlaender, H.L.: On disjoint cycles. International Journal of Foundations of Computer Science 5(1), 59–68 (1994)
7. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. Journal of Computer and System Sciences 75(8), 423–434 (2009)
8. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (Meta) kernelization. In: 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009. pp. 629–638. IEEE Computer Society (2009)
9. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. Theoretical Computer Science 412(35), 4570–4578 (2011)
10. Brouwer, A.E., Veldman, H.J.: Contractibility and NP-completeness. Journal of Graph Theory 11(1), 71–79 (1987)
11. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Information Processing Letters 58(4), 171–176 (1996)
12. Cao, Y., Chen, J., Liu, Y.: On feedback vertex set new measure and new structures. In: Kaplan, H. (ed.) 12th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2010. Lecture Notes in Computer Science, vol. 6139, pp. 93–104. Springer (2010)
13. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for feedback vertex set problems. Journal of Computer and System Sciences 74(7), 1188–1198 (2008)
14. Chen, J., Kneis, J., Lu, S., Molle, D., Richter, S., Rossmanith, P., Sze, S.H., Zhang, F.: Randomized divide-and-conquer: Improved path, matching, and packing algorithms. SIAM Journal on Computing 38(6), 2526–2547 (2009)
15. Courcelle, B.: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. Information and Computation 85(1), 12–75 (1990)
16. Cygan, M.: Deterministic parameterized connected vertex cover. In: Fomin, F. (ed.) 13th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2012. Lecture Notes in Computer Science, Springer (to appear), manuscript arXiv:1202.6642, February 2012
17. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: Ostrovsky, R. (ed.) 52nd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2011. pp. 150–159. IEEE (2011)
18. Dehne, F.K.H.A., Fellows, M.R., Langston, M.A., Rosamond, F.A., Stevens, K.: An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. Theory of Computing Systems 41(3), 479–492 (2007)
19. Díaz, J., Thilikos, D.M.: Fast FPT-algorithms for cleaning grids. In: Durand, B., Thomas, W. (eds.) 23rd Annual Symposium on Theoretical Aspects of Computer Science, STACS 2006. Lecture Notes in Computer Science, vol. 3884, pp. 361–371. Springer (2006)
20. Dom, M., Lokshtanov, D., Saurabh, S.: Incompressibility through colors and IDs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S.E., Thomas, W. (eds.) 36th International Colloquium on Automata, Languages and Programming, ICALP 2009. Lecture Notes in Computer Science, vol. 5555, pp. 378–389. Springer (2009)
21. Downey, R.G., Fellows, R.: Parameterized Complexity. Monographs in computer science, Springer (1999)

22. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science, Springer (2006)
23. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. Journal of Computer and System Sciences 77(1), 91–106 (2011)
24. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co., New York, NY, USA (1979)
25. Golovach, P.A., Kaminski, M., Paulusma, D., Thilikos, D.M.: Increasing the minimum degree of a graph by contractions. In: Marx, D., Rossmanith, P. (eds.) 6th International SymposiumP on arameterized and Exact Computation, IPEC 2011. Lecture Notes in Computer Science, vol. 7112, pp. 67–79. Springer (2011)
26. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. Journal of Computer and System Sciences 72(8), 1386–1396 (2006)
27. Heggernes, P., van 't Hof, P., Jansen, B.M.P., Kratsch, S., Villanger, Y.: Parameterized complexity of vertex deletion into perfect graph classes. In: Owe, O., Steffen, M., Telle, J.A. (eds.) 18th International Symposium on Fundamentals of Computation Theory, FCT 2011. Lecture Notes in Computer Science, vol. 6914, pp. 240–251. Springer (2011)
28. Heggernes, P., van 't Hof, P., Lévêque, B., Lokshtanov, D., Paul, C.: Contracting graphs to paths and trees. In: Marx, D., Rossmanith, P. (eds.) 6th International Symposium on arameterized and Exact Computation, IPEC 2011. Lecture Notes in Computer Science, vol. 7112, pp. 55–66. Springer (2011)
29. Heggernes, P., van 't Hof, P., Lévêque, B., Paul, C.: Contracting chordal graphs and bipartite graphs to paths and trees. Electronic Notes in Discrete Mathematics 37, 87–92 (2011)
30. Heggernes, P., van 't Hof, P., Lokshtanov, D., Paul, C.: Obtaining a bipartite graph by contracting few edges. In: Chakraborty, S., Kumar, A. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011. LIPIcs, vol. 13, pp. 217–228. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2011)
31. van 't Hof, P., Villanger, Y.: Proper interval vertex deletion. Algorithmica (to appear), http://dx.doi.org/10.1007/s00453-012-9661-3
32. Kawarabayashi, K., Reed, B.A.: An (almost) linear time algorithm for odd cyles transversal. In: Charikar, M. (ed.) 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010. pp. 365–378 (2010)
33. Martin, B., Paulusma, D.: The computational complexity of disconnected cut and 2k 2-partition. In: Lee, J.H.M. (ed.) 17th International Conference on Principles and Practice of Constraint Programming, CP 2011. Lecture Notes in Computer Science, vol. 6876, pp. 561–575. Springer (2011)
34. Marx, D.: Chordal deletion is fixed-parameter tractable. Algorithmica 57(4), 747–768 (2010)
35. Marx, D., Schlotter, I.: Obtaining a planar graph by vertex deletion. Algorithmica 62(3-4), 807–822 (2012)
36. Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In: 36th Annual Symposium on Foundations of Computer Science, FOCS 1995. pp. 182–191. IEEE Computer Society (1995)
37. Natanzon, A., Shamir, R., Sharan, R.: Complexity classification of some edge modification problems. Discrete Applied Mathematics 113(1), 109–128 (2001)
38. Philip, G., Raman, V., Villanger, Y.: A quartic kernel for pathwidth-one vertex deletion. In: Thilikos, D.M. (ed.) 36th International Workshop on Graph Theoretic

Concepts in Computer Science, WG 2010. Lecture Notes in Computer Science, vol. 6410, pp. 196–207 (2010)
39. Pilipczuk, M.: Problems parameterized by treewidth tractable in single exponential time: A logical approach. In: Murlak, F., Sankowski, P. (eds.) 36th International Symposium on Mathematical Foundations of Computer Science 2011, MFCS 2011. Lecture Notes in Computer Science, vol. 6907, pp. 520–531. Springer (2011)
40. Thomassé, S.: A $4k^2$ kernel for feedback vertex set. ACM Transactions on Algorithms 6(2) (2010)
41. Watanabe, T., Ae, T., Nakamura, A.: On the removal of forbidden graphs by edge-deletion or by edge-contraction. Discrete Applied Mathematics 3(2), 151 – 153 (1981)
42. Watanabe, T., Ae, T., Nakamura, A.: On the NP-hardness of edge-deletion and -contraction problems. Discrete Applied Mathematics 6(1), 63 – 78 (1983)
43. Yannakakis, M.: Node- and edge-deletion NP-complete problems. In: Lipton, R.J., Burkhard, W.A., Savitch, W.J., Friedman, E.P., Aho, A.V. (eds.) 10th Annual ACM Symposium on Theory of Computing, STOC 1978. pp. 253–264. ACM (1978)
44. Yannakakis, M.: The effect of a connectivity requirement on the complexity of maximum subgraph problems. Journal of the ACM 26(4), 618–630 (1979)
45. Yannakakis, M.: Edge-deletion problems. SIAM Journal on Computing 10(2), 297–309 (1981)