Bandwidth and Distortion Revisited

Marek Cygan          Marcin Pilipczuk∗

Abstract

In this paper we merge recent developments on exact algorithms for finding an ordering of vertices of a given graph that minimizes bandwidth (the BANDWIDTH problem) and for finding an embedding of a given graph into a line that minimizes distortion (the DISTORTION problem). For both problems we develop algorithms that work in $O(9.363^n)$ time and polynomial space. For BANDWIDTH, this improves $O^*(10^n)$ algorithm by Feige and Kilian from 2000, for DISTORTION this is the first polynomial space exact algorithm that works in $O(c^n)$ time we are aware of. As a coproduct, we enhance the $O(5n+o(n))$–time and $O^*(2^n)$–space algorithm for DISTORTION by Fomin et al. to an algorithm working in $O(4.383^n)$ time and space.

1 Introduction

Recently the NP–complete BANDWIDTH problem, together with a similar problem of embedding a graph into a real line with minimal distortion (called DISTORTION in this paper), attracted some attention from the side of exact (and therefore not polynomial) algorithms.

Given a graph $G$ with $n$ vertices, an ordering is a bijective function $\pi : V(G) \to \{1, 2, \ldots, n\}$. Bandwidth of $\pi$ is a maximal length of an edge, i.e., $bw(\pi) = \max_{uv \in E(G)} |\pi(u) - \pi(v)|$. The BANDWIDTH problem, given a graph $G$ and a positive integer $b$, asks if there exists an ordering of bandwidth at most $b$.

Given a graph $G$, an embedding of $G$ into a real line is a function $\pi : G \to \mathbb{R}$. For every pair of distinct vertices $u, v \in V(G)$ we define a distortion of $u$ and $v$ by $\text{dist}(u, v) = |\pi(u) - \pi(v)|/d_G(u, v)$, where $d_G$ denotes the distance in the graph $G$. A contraction and an expansion of $\pi$, denoted $\text{contr}(\pi)$ and $\text{expan}(\pi)$ respectively, are the minimal and maximal distortion over all pairs of distinct vertices in $V(G)$. The distortion of $\pi$, denoted $\text{dist}(\pi)$, equals to $\text{expan}(\pi)/\text{contr}(\pi)$. The DISTORTION problem, given a graph $G$ and a positive real number $d$, asks if there exists an embedding with distortion at most $d$. Note that the distortion of an embedding does not change if we change $\pi$ infinitely, and we can rescale $\pi$ by $1/\text{contr}(\pi)$ and obtain $\pi$ with contraction exactly 1. Therefore, in this paper, we limit ourselves only to embeddings with contraction at least 1 and we optimize the expansion of the embedding, that is, we try to construct embeddings with contraction at least 1 and with expansion at most $d$.

The first non–trivial exact algorithm for the BANDWIDTH problem was developed by Feige and Kilian in 2000 [6]. It works in polynomial space and $O^*(10^n)$ time. Recently we improved the time bound to $O^*(5^n)$ [4], $O(4.83^n)$ [3] and $O^*(20^{n/2})$ [5]. However, the cost of the improvements was exponential space complexity: $O^*(2^n)$, $O^*(4^n)$, $O^*(20^{n/2})$ respectively. In

∗Dept. of Mathematics, Computer Science and Mechanics, University of Warsaw, Poland, {cygan, malcin}@mimuw.edu.pl
In this paper we present a link between aforementioned results and develop an \( O(n^d(2d+1)^{2d}) \) time, which does not reach the \( O(c^n) \) complexity for \( d = \Omega(n) \).

In this paper we present a link between aforementioned results and develop a \( O(9.363^n) \)–time and polynomial space algorithm for both BANDWIDTH and DISTORTION. First, we develop a \( O(4.383^n) \)–time and space algorithm for DISTORTION, using ideas both from \( O^*(2^n/2) \) algorithm for BANDWIDTH \([5]\) and \( O(5^{n+o(n)}) \) algorithm for DISTORTION \([9]\). Then, we use an approach somehow similar to those of Feige and Kilian \([6]\) to reduce space to polynomial, at the cost of time complexity, obtaining the aforementioned algorithms. We are not aware of any exact polynomial–space algorithms that work in \( O(c^n) \) time for DISTORTION or are faster than Feige and Kilian’s algorithm for BANDWIDTH.

In Section 2 we gather results on partial bucket functions: tool that was used in all previous algorithms for DISTORTION and BANDWIDTH. In Section 3 we recall the \( O^*(2^n/2) \) algorithm \([5]\) and show how to transform it into \( O(9.363^n) \)–time and polynomial space algorithm for BANDWIDTH. Section 4 is devoted to DISTORTION: first, we merge ideas from \([4]\) and \([9]\) to obtain an \( O^*(4.383^n) \)–time and space algorithm for DISTORTION. Then we apply the same trick as for BANDWIDTH to obtain an \( O(9.363^n) \)–time and polynomial space algorithm.

In the following sections we assume that we are given a connected undirected graph \( G = (V,E) \) with \( n = |V| \). Note that BANDWIDTH trivially decomposes into subproblems on connected components, whereas answer to DISTORTION is always negative for a disconnected graph. Proofs of results marked with ♣ are postponed to Appendix B.

2 Partial bucket functions

In this section we gather results on partial bucket functions, a tool used in algorithms for both BANDWIDTH and DISTORTION. Most ideas here are based on the \( O^*(2^n/2) \) algorithm for BANDWIDTH \([5]\).

**Definition 2.1.** A partial bucket function is a pair \( (A, f) \), such that \( A \subseteq V \), \( f : A \rightarrow \mathbb{Z} \) and there exists \( \bar{f} : V \rightarrow \mathbb{Z} \) satisfying:

1. \( \bar{f}|_A = f \);
2. if \( uv \in E \) then \( |\bar{f}(u) - \bar{f}(v)| \leq 1 \), in particular, if \( u, v \in A \) then \( |f(u) - f(v)| \leq 1 \);
3. if \( uv \in E \), \( u \in A \) and \( v \notin A \) then \( \bar{f}(u) \geq \bar{f}(v) \), i.e., \( \bar{f}(u) = \bar{f}(v) \) or \( \bar{f}(u) = \bar{f}(v) + 1 \).

We say that such a function \( \bar{f} \) is a bucket extension of \( f \).

**Definition 2.2.** Assume we have two partial bucket functions \( (A, f) \) and \( (A', f') \) such that \( A' = A \cup \{v\}, v \notin A \) and \( f|_A = f \), we say that \( (A', f') \) is a successor of \( (A, f) \) with vertex \( v \) if there does not exist any \( uv \in E \), \( u \in A \) such that \( f(u) < f'(v) \).

\(^1\) The complexity analysis of our algorithm, in particular the proof in Appendix A proves that the algorithm from \([5]\) works in \( O(4.383^n) \) time and space too. However, we do not state it as a new result in this paper, since analysis based on this approach will be published in the journal version of \([5]\).
Lemma 2.3. Assume that $A \subseteq V$ and $f : A \to \mathbb{Z}$. Moreover, let $A \subseteq B \subseteq V$, $f' : B \to \mathbb{Z}$ and $f'|_A = f$. Then one can find in polynomial time a bucket extension $\tilde{f}$ of $f$ such that $\tilde{f}|_B = f'$ or state that such bucket extension does not exist.

Proof. The case $A = B = \emptyset$ is trivial, so we may assume there exists some $v_0 \in B$. W.l.o.g. we may assume $f'(v_0) = 0$. Therefore any valid bucket extension should satisfy $\tilde{f}(V) \subseteq \{-n, -n+1, \ldots, n\}$.

We calculate for every $v \in V \setminus A$ the value $p(v) \subseteq \{-n, -n+1, \ldots, n\}$, intuitively, the set of possible values for $\tilde{f}(v)$, by the following algorithm.

Algorithm 2.1 Calculate values $p(v)$ — the sets of valid values for $\tilde{f}(v)$.

1. Set $p(v) := \{-n, -n+1, \ldots, n\}$ for all $v \in V \setminus B$.
2. Set $p(v) := \{f'(v)\}$ for all $v \in B \setminus A$.
3. repeat
4. for all $v \in V \setminus B$ do
5. \hspace{1em} $p(v) := p(v) \cap \bigcap_{u \in N(v) \cap A} (f(u) - 1, f(u)]) \cap \bigcap_{u \in N(v) \setminus A} \bigcup_{i \in \mathbb{Z}} \{i - 1, i, i + 1\}$
6. until some $p(v)$ is empty or we do not change any $p(v)$ in the inner loop
7. return True iff all $p(v)$ remain nonempty.

To prove that Algorithm 2.1 correctly checks if there exists a valid bucket extension $\tilde{f}$ note the following:

1. Let $\tilde{f}$ be a bucket extension of $(A, f)$ such that $\tilde{f}|_B = f'$. Then, at every step of the algorithm $\tilde{f}(v) \in p(v)$ for every $v \in V \setminus A$.
2. If the algorithm returns nonempty $p(v)$ for every $v \in V \setminus A$, setting $\tilde{f}(v) = \min p(v)$ constructs a valid bucket extension of $(A, f)$. Moreover, since we start with $p(v) = \{f'(v)\}$ for $v \in B \setminus A$, we obtain $\tilde{f}|_B = f'$.

\hfill $\square$

Corollary 2.4. One can check in polynomial time whether a given pair $(A, f)$ is a partial bucket function. Moreover one can check whether $(A', f')$ is a successor of $(A, f)$ in polynomial time too.

Proof. To check if $(A, f)$ is a partial bucket function we simply run the algorithm from Lemma 2.3 for $B = A$ and $f' = f$. Conditions for being a successor of $(A, f)$ are trivial to check. \hfill $\square$

Lemma 2.5. Let $N \in \mathbb{Z}_+$. Then there are at most $2N \cdot 5^{n-1}$ triples $(A, f, \tilde{f})$ such that $(A, f)$ is a partial bucket function and $\tilde{f}$ is a bucket extension of $f$ satisfying $\tilde{f}(V) \subseteq \{1, 2, \ldots, N\}$.

Proof. Note that if $(A, f)$ is a partial bucket function in the graph $G$ and $\tilde{f}$ is a bucket extension, and $G'$ is a graph created from $G$ by removing an edge, then $(A, f)$ and $\tilde{f}$ remain partial bucket function and its bucket extension in $G'$. Therefore we may assume that $G$ is a tree, rooted at $v_r$.

There are $2N$ possibilities to choose the value of $\tilde{f}(v_r)$ and whether $v_r \in A$ or $v_r \notin A$. We now construct all interesting triples $(A, f, \tilde{f})$ in a root-to-leaves order in the graph $G$. If we are at a node $v$ with its parent $w$, then $\tilde{f}(v) \in \{f(w) - 1, f(w), f(w) + 1\}$. However, if $w \in A$ then we cannot both have $f(v) = f(w) + 1$ and $v \notin A$. Similarly, if $w \notin A$ then we cannot both
have \( f(v) = f(w) - 1 \) and \( v \in A \). Therefore we have 5 options to choose \( f(v) \) and whether \( v \in A \) or \( v \notin A \). Finally, we obtain at most \( 2N \cdot 5^{a-1} \) triples \((A, f, \bar{f})\).

**Lemma 2.6 (♣).** Let \((A, f)\) be a partial bucket function. Then all bucket extensions of \( f \) can be generated with a polynomial delay, using polynomial space.

The proof of the theorem below is an adjusted and improved proof of a bound of the number of states in the \( O^*(20^{n/2}) \) algorithm for BANDWIDTH \( [5] \). The proof can be found in Appendix A.

**Theorem 2.7.** Let \( N \in \mathbb{Z}_+ \). There exists a constant \( c < 4.383 \) such that there are \( O(N \cdot c^n) \) partial bucket functions \((A, f)\) such that there exists a bucket extension \( \bar{f} \) satisfying \( \bar{f}(V) \subseteq \{1, 2, \ldots, N\} \). Moreover, all such partial bucket functions can be generated in \( O^*(N \cdot c^n) \) time using polynomial space.

### 3 Poly-space algorithm for BANDWIDTH

In this section we describe an \( O(9.363^n) \)-time and polynomial-space algorithm solving BANDWIDTH. As an input, the algorithm takes a graph \( G = (V, E) \) with \(|V| = n\) and an integer \( 1 \leq b < n \) and decides, whether \( G \) has an ordering with bandwidth at most \( b \).

#### 3.1 Preliminaries

First, let us recall some important observations made in \([4]\). An ordering \( \pi \) is called a \( b \)-ordering if \( \text{bw} (\pi) \leq b \). Let \( \text{Pos} = \{1, 2, \ldots, n\} \) be the set of possible positions and for every position \( i \in \text{Pos} \) we define the segment it belongs to by \( \text{segment}(i) = \left\lceil \frac{i}{b+1} \right\rceil \) and the color of it by \( \text{color}(i) = (i - 1) \mod (b + 1) + 1 \). By \( \text{Seg} = \{1, 2, \ldots, \left\lceil \frac{n}{b+1} \right\rceil \} \) we denote the set of possible segments, and by \( \text{Col} = \{1, 2, \ldots, b + 1\} \) the set of possible colors. The pair \((\text{color}(i), \text{segment}(i))\) defines the position \( i \) uniquely. We order positions lexicographically by pairs \((\text{color}(i), \text{segment}(i))\), i.e., the color has higher order that the segment number, and call this order the color order of positions. By \( \text{Pos}_i \), we denote the set of the first \( i \) positions in the color order. Given some (maybe partial) ordering \( \pi \), and \( v \in V \) for which \( \pi(v) \) is defined, by \( \text{color}(v) \) and \( \text{segment}(v) \) we understand \( \text{color}(\pi(v)) \) and \( \text{segment}(\pi(v)) \) respectively.

Let us recall the crucial observation made in \([4]\).

**Lemma 3.1 ([4], Lemma 8).** Let \( \pi \) be an ordering. It is a \( b \)-ordering iff, for every \( uv \in E \), \(|\text{segment}(u) - \text{segment}(v)| \leq 1 \) and if \( \text{segment}(u) + 1 = \text{segment}(v) \) then \( \text{color}(u) > \text{color}(v) \) (equivalently, \( \pi(u) \) is later in color order than \( \pi(v) \)).

#### 3.2 \( O^*(20^{n/2}) \) algorithm from \([5]\)

First let us recall the \( O^*(20^{n/2}) \)-time and space algorithm from \([5]\).

**Definition 3.2.** A state is a partial bucket assignment \((A, f)\) such that the multiset \( \{f(v) : v \in A\} \) is equal to the multiset \( \{\text{segment}(i) : i \in \text{Pos}_A\} \). A state \((A \cup \{v\}, f')\) is a successor of a state \((A, f)\) with a vertex \( v \notin A \) if \((A \cup \{v\}, f')\) as a partial bucket function is a successor of a partial bucket function \((A, f)\).

**Theorem 3.3 ([5], Lemmas 16 and 17).** 1. Let \( \pi \) be a \( b \)-ordering. For \( 0 \leq k \leq n \) let \( A_k = \{v \in V : \pi(v) \in \text{Pos}_k\} \) and \( f_k = \text{segment}|_{A_k} \). Then every \((A_k, f_k)\) is a state and for every \( 0 \leq k < n \) the state \((A_{k+1}, f_{k+1})\) is a successor of the state \((A_k, f_k)\).
2. Assume we have states \((A_k, f_k)\) for \(0 \leq k \leq n\) and for all \(0 \leq k < n\), the state \((A_{k+1}, f_{k+1})\) is a successor of the state \((A_k, f_k)\) with the vertex \(v_{k+1}\). Let \(\pi\) be an ordering assigning \(v_k\) to the \(k\)-th position in the color order. Then \(\pi\) is a \(b\)-ordering.

The algorithm of [5] works as follows: we do a depth-first search from the state \((\emptyset, \emptyset)\) and seek for a state \((V, \cdot)\). At a state \((A, f)\) we generate in polynomial time all successors of this state and memoize visited states. Theorem 3.3 implies that we reach state \((V, \cdot)\) iff there exists a \(b\)-ordering. Moreover, Theorem 2.7 (with \(N = n\)) implies that we visit at most \(O(4^{(3.83n)})\) states; generating all successors of a given state can be done in polynomial time due to Corollary 2.4 so the algorithm works in \(O(4^{(3.83n)})\) time and space.

3.3 The \(O(9.363^n)\)-time and polynomial space algorithm

**Lemma 3.4.** Let \((A, f)\) and \((B, g)\) be a pair of states such that \(A \subseteq B\) and \(g|_A = f\). Let \(a = |A|\) and \(b = |B|\). Then one can check in \(O^*(4^{b-a})\)-time and polynomial space if there exists a sequence of states \((A_i, f_i) = (A_a, f_a), (A_{a+1}, f_{a+1}), \ldots, (A_b, f_b) = (B, g)\) such that \((A_{i+1}, f_{i+1})\) is an successor of \((A_i, f_i)\) for \(a \leq i < b\).

**Proof.** First note that a set \(A_i\) determines the function \(f_i\), since \(f_i = g|_{A_i}\). Let \(m := b - a\). If \(m = 1\), we need to check only if \((B, g)\) is a successor of \((A, f)\), what can be done in polynomial time. Otherwise, let \(k = \lfloor \frac{a+b}{m} \rfloor\) and guess \(A_k\): there are roughly \(2^m\) choices. Set \(f_k = g|_{A_k}\). Recursively, check if there is a path of states from \((A, f)\) to \((A_k, f_k)\) and from \((A_k, f_k)\) to \((B, g)\).

The algorithm clearly works in polynomial space; now let us estimate the time it consumes. At one step, it does some polynomial computation and invokes roughly \(2^{m+1}\) times itself recursively for \(b - a \sim m/2\). Therefore it works in \(O^*(4^m)\) time. \(\square\)

Let \(\alpha = 0.5475\). The algorithm works in the same fashion as in [5]: it seeks for a path of states \((A_i, f_i)\) from \((\emptyset, \emptyset)\) to \((V, \cdot)\) such that \((A_{i+1}, f_{i+1})\) is a successor of \((A_i, f_i)\) for \(0 \leq i < n\). However, since we are limited to polynomial space, we cannot do a simple search. Instead, we guess middle states on the path, similarly as in Lemma 3.4. The algorithm works as follows:

1. Let \(k := \lfloor \alpha n \rfloor\) and guess the state \((A_k, f_k)\). By Theorem 2.7 with \(N = n\), we can enumerate all partial bucket functions in \(O(4(3.83n))\). We enumerate them and drop those that are not states or have the size of the domain different than \(k\).

2. Using Lemma 3.4, check if there is a path of states from \((\emptyset, \emptyset)\) to \((A_k, f_k)\). This phase works in time \(4^m\). In total, for all \((A_k, f_k)\), this phase works in time \(O(4^{(3.83n).4^m}) = O(9.363^n)\).

3. Guess the state \((V, f_n)\): \(f_n\) needs to be a bucket extension of the partial bucket function \((A_k, f_k)\). By Lemma 2.6, bucket extensions can be enumerated with polynomial delay; we simply drop those that are not states. By Lemma 2.5 with \(N = n\), there will be at most \(O^*(5^n)\) pairs of states \((A_k, f_k)\) and \((V, f_n)\).

4. Using Lemma 3.4, check if there is a path from the state \((A_k, f_k)\) to \((V, f_n)\). This phase works in time \(O^*(4^{(1-\alpha)n})\). In total, for all \((A_k, f_k)\) and \((V, f_n)\), this phase works in time \(O^*(5^n.4^{(1-\alpha)n}) = O(9.363^n)\).

5. Return true, if for any \((A_k, f_k)\) and \((V, f_n)\) both applications of Lemma 3.4 return success.

Theorem 3.3 ensures that the algorithm is correct. In memory we keep only states \((A_k, f_k)\), \((V, f_n)\), recursion stack generated by the algorithm from Lemma 3.4 and state of generators of
states \((A_k, f_k)\) and \((V, f_n)\), so the algorithm works in polynomial space. Comments above prove that it consumes at most \(O(9.363^n)\) time.

4 Algorithms for Distortion

We consider algorithms that, given a connected graph \(G\) with \(n\) vertices, and positive real number \(d\) decides if \(G\) can be embedded into a line with distortion at most \(d\). First, let us recall the basis of the approach of Fomin et al. [9]. Recall that \(d_G(u, v)\) denotes the distance between vertices \(u\) and \(v\) in the graph \(G\).

**Definition 4.1.** Given an embedding \(\pi : V \rightarrow \mathbb{Z}\), we say that \(v\) pushes \(u\) iff \(d_G(u, v) = |\pi(u) - \pi(v)|\). An embedding is called pushing, if \(V = \{v_1, v_2, \ldots, v_n\}\) and \(\pi(v_1) < \pi(v_2) < \ldots < \pi(v_n)\) then \(v_i\) pushes \(v_{i+1}\) for all \(1 \leq i < n\).

**Lemma 4.2 ([7]).** If \(G\) can be embedded into the line with distortion \(d\), then there is a pushing embedding of \(G\) into the line with distortion \(d\). Every pushing embedding of \(G\) into the line has contraction at least 1. Moreover, let \(\pi\) be a pushing embedding of a connected graph \(G\) into the line with distortion at most \(d\) and let \(V = \{v_1, v_2, \ldots, v_n\}\) be such an ordering \(\pi\) that \(\pi(v_1) < \pi(v_2) < \ldots < \pi(v_n)\). Then \(\pi(v_{i+1}) - \pi(v_i) \leq d\) for all \(1 \leq i < n\).

Therefore, we only consider pushing embeddings and hence assume that \(d\) is a positive integer. Note that a pushing embedding of a connected graph of at least 2 vertices has contraction exactly 1, since \(d_G(v_1, v_2) = |\pi(u_2) - \pi(u_1)|\). Therefore distortion equals expansion. As any connected graph with \(n\) vertices can be embedded into a line with distortion at most \(2n - 1\) [1], this decisive approach suffices to find the minimal distortion of \(G\).

We may assume that \(\pi(V) \subseteq \{1, 2, \ldots, n(d + 1)\}\). Now, let us introduce the concept of segments, adjusted for the Distortion problem. Here the set of available positions is \(\text{Pos} = \{1, 2, \ldots, n(d + 1)\}\) and a segment of a position \(i\) is \(\text{segment}(i) = \left\lfloor \frac{i}{d+1} \right\rfloor\), i.e., a \(j\)-th segment is an integer interval of the form \(\{(j-1)(d+1) + 1, (j-1)(d+1) + 2, \ldots, j(d+1)\}\).

The color of a position is \(\text{color}(i) = (i - 1) \mod (d + 1) + 1\). By \(\text{Seg} = \{1, 2, \ldots, n\}\) we denote the set of possible segments and by \(\text{Col} = \{1, 2, \ldots, d + 1\}\) the set of possible colors. The pair \((\text{color}(i), \text{segment}(i))\) defines the position \(i\) uniquely. We order the positions lexicographically by pairs \((\text{color}(i), \text{segment}(i))\) and call this order color order of positions. By \(\text{Pos}\), we denote the set of the first \(i\) positions in the color order and by \(\text{Seg}_i\) we denote the set of positions in the \(i\)-th segment. Given some, maybe partial, embedding \(\pi\), by \(\text{color}(v)\) and \(\text{segment}(v)\) we denote \(\text{color}(\pi(v))\) and \(\text{segment}(\pi(v))\) respectively.

Similarly as in the case of Bandwidth, the following equivalence holds (cf. Lemma 3.1).

**Lemma 4.3 ([9]).** Let \(\pi\) be a pushing embedding. Then \(\pi\) has distortion at most \(d\) if and only if for every \(uv \in E\), \(|\text{segment}(u) - \text{segment}(v)| \leq 1\) and if \(\text{segment}(u) + 1 = \text{segment}(v)\) then \(\text{color}(u) > \text{color}(v)\), i.e., \(\pi(u)\) is later in the color order than \(\pi(v)\).

Similarly as in [9], we solve the following extended case of Distortion as a subproblem. As an input to the subproblem, we are given an induced subgraph \(G[X]\) of \(G\), an integer \(r\) (called the number of segments), a subset \(Z \subseteq X\) and a function \(\pi : Z \rightarrow \text{Seg}_0 \cup \text{Seg}_{r+1}\). Given this input, we ask, if there exists a pushing embedding \(\pi : X \rightarrow \{-d, -d + 1, \ldots, r(d + 1)\}\) with distortion at most \(d\) such that \(\pi|_Z = \pi, \pi(X \setminus Z) \subseteq \{1, 2, \ldots, r(d + 1)\}\). Moreover, we demand that \(\pi\) does not leave any empty segment, i.e., for every \(1 \leq i \leq r, \pi^{-1}(\text{Seg}_i) \neq \emptyset\).
Theorem 4.4. The extended Distortion problem can be solved in $O(1.383^{|X|} |Z|_n^{O(r)})$ time and space. If we are restricted to polynomial space, the extended Distortion problem can be solved in $O(9.36^{|X|} |Z|^{O(r \log n)})$ time.

Let $n = |X \setminus Z|$. The algorithm for Theorem 4.4 goes as follows. First, for each segment $1 \leq i \leq r$ we guess the vertex $v_i$ and position $1 \leq p_i \leq r(d+1)$ such that $\text{Seg}(p_i) = i$. There are roughly $O(n^{O(r)})$ possible guesses (if $r > n$ the answer is immediately negative). We seek for embeddings $\pi$ such that for every $1 \leq i \leq r$ position $\pi(v_i) = p_i$, and there is no vertex assigned to any position in the segment $i$ with color earlier than $\text{color}(p_i)$, i.e., $v_i$ is the first vertex in the segment $i$. If there exists $z \in Z$ such that $\pi(z) \leq 0$, then we require that $v_i$ is pushed by such $z$ that $\pi(z)$ is the largest nonpositive possible.

Along the lines of the algorithm for Bandwidth [5] and algorithm for Distortion by Fomin et al. [9], we define state and a state successor as follows:

Definition 4.5. A state is a triple $(p, (A, f), (H, h))$ such that:
1. $0 \leq p \leq r(d+1)$ is an integer,
2. $(A, f)$ is a partial bucket function,
3. $H \subseteq A$ is a set of vertices such that $H \cap \text{Seg}_i$ is nonempty iff $f^{-1}(i)$ is nonempty,
4. $h : H \to \text{Pos}_p$ and if $v \in H$ then $f(v) = \text{segment}(h(v))$,
5. if for any segment $1 \leq i \leq r$, vertex $v_i \in H$, then $h(v_i) = p_i$,
6. if for any segment $1 \leq i \leq r$ position $p_i \in \text{Pos}_i$ then $v_i \in A$ and $f(v_i) = i$.

Definition 4.6. We say that a state $(p+1, (A_2, f_2), (H_2, h_2))$ is a successor of a state $(p, (A_1, f_1), (H_1, h_1))$ iff:
1. $A_2 = A_1$ or $A_2 = A_1 \cup \{v\}$,
2. if $A_2 = A_1$ then $f_2 = f_1$, $H_1 = H_2$ and $h_1 = h_2$.
3. if $A_2 = A_1 \cup \{v\}$, then:
   (a) partial bucket function $(A_2, f_2)$ is a successor of the partial bucket function $(A_1, f_1)$ with the vertex $v$, such that $f_2(v) = \text{segment}(p+1)$,
   (b) $H_2 = (H_1 \setminus f_1^{-1}(\text{segment}(p+1))) \cup \{v\}$,
   (c) $h_2 = h_1 |_{H_1 \cap H_2} \cup (v, p+1)$,
   (d) if $H_1 \cap f_1^{-1}(\text{segment}(p+1)) = \{w\}$, then $d_G(v, w) = h_2(v) - h_1(w)$,
   (e) for any $z \in Z$, $d_G(z, v) \leq |\pi(z) - (p+1)| \leq d \cdot d_G(z, v)$.

Definition 4.7. We say that a state $(r(d+1), (V, f), (H, h))$ is a final state iff for each segment $1 \leq i \leq r$ we have $\{w_i\} = H \cap \text{Seg}_i$ (i.e., $H \cap \text{Seg}_i$ is nonempty), $w_i$ pushes $v_{i+1}$ for $i < r$ and $w_r$ pushes first $z \in Z$ such that $\pi(z) \in \text{Seg}_{r+1}$ (if such $z$ exists).

The following equivalence holds:

Lemma 4.8. Let $\pi$ be a pushing embedding and a solution to the extended Distortion problem with distortion at most $d$. Assume that $\pi(v_i) = p_i$ and this is the first vertex in the segment $i$ for every segment $1 \leq i \leq r$, i.e., the initial guesses are correct with respect to the solution $\pi$. For each $1 \leq p \leq r(d+1)$ we define $(A_p, f_p)$ and $(H_p, h_p)$ as follows:
1. $A_p = \pi^{-1}((\text{Pos}_p)$ and $f_p = \text{segment} |_{A_p}$,
2. For each segment \( 1 \leq i \leq r \) we take \( \pi_i \) as the vertex in \( \pi^{-1}(\text{Pos}_p \cap \text{Seg}_i) \) with the greatest color of position and take \( \pi(w_i) = \pi_i(w_i) \). If \( \pi^{-1}(\text{Pos}_p \cap \text{Seg}_i) = \emptyset \), we take \( H_p \cap \text{Seg}_i = \emptyset \).

Then \( S_p = (p, (A_p, f_p), (H_p, h_p)) \) is a state and \( S_{p+1} = (p+1, (A_{p+1}, f_{p+1}), (H_{p+1}, h_{p+1})) \) is its successor if \( p < r(d+1) \). Moreover, \( S_{r(d+1)} \) is a final state.

Proof: First note that, similarly as in the case of \( \text{BANDWIDTH} \), \( (A_p, f_p) \) is a partial bucket function and \( (A_{p+1}, f_{p+1}) \) is a successor of \( (A_p, f_p) \). Indeed, the conditions for a partial bucket function and its successor are implied by Lemma 4.3.

The check that \( (H_p, h_p) \) satisfies the conditions for being a state is straightforward. Let us now look at the conditions for the successor. The only nontrivial part is that if in \( H_p \) the vertex \( w \) is replaced by \( v \) in \( H_{p+1} \), then \( d_G(v, w) = h_{p+1}(v) - h_{p}(w) \). However, this is implied by the fact that \( \pi \) is a pushing embedding.

To see that \( S_{r(d+1)} \) is a final state recall that \( \pi \) leaves no segment \( \text{Seg}_i, 1 \leq i \leq r \), nonempty and it is a pushing embedding.

Lemma 4.9. Assume that we have a sequence of states \( (S_p)_{p=0}^{r(d+1)} \), \( S_p = (p, (A_p, f_p), (H_p, h_p)) \) such that \( S_{p+1} \) is a successor of \( S_p \) for \( 0 \leq p < r(d+1) \) and \( S_{r(d+1)} \) is a final state. Let \( \pi = \bigcup_{p=0}^{r(d+1)} h_p \). Then \( \pi \) is a solution to the extended \( \text{DISTORTION} \) problem with distortion at most \( d \). Moreover, \( \pi(v_i) = p_i \) for all \( 1 \leq i \leq r \).

Proof. Note that the conditions for the final state imply that \( \pi \) leaves every segment from 1 to \( r \) nonempty. Moreover, the conditions for \( (H_p, h_p) \) imply that \( \pi(v_i) = p_i \) and \( v_i \) is the first vertex assigned in segment \( i \).

First we check if \( \pi \) is a pushing embedding. Let \( v \) and \( w \) be two vertices such that \( \pi(v) < \pi(w) \) and there is no \( u \) with \( \pi(v) < \pi(u) < \pi(w) \). If \( \text{segment}(v) = \text{segment}(w) \), then \( \pi(w) - \pi(v) = d_G(v, w) \) is ensured by the state successor definition at step, where \( S_{p+1} \) is a successor of the state \( S_p \) with the vertex \( w \). Otherwise, if \( \text{segment}(v) + 1 = \text{segment}(w) \), then \( w = v_{\text{segment}(v)} \) or \( w \) is the first vertex of \( Z \) in segment \( r + 1 \) and the fact that \( v \) pushes \( w \) is implied by the condition of the final state. The possibility that \( \text{segment}(v) + 1 < \text{segment}(w) \) is forbidden since in the final state \( H_{r(d+1)} \cap \text{Seg}_i \neq \emptyset \) for \( 1 \leq i \leq r \).

Now we check if for each edge \( uv, |\pi(u) - \pi(v)| \leq d \). Assume not, let \( \pi(u) + d < \pi(v) \) and let \( S_k \) be a successor of the state \( S_{k-1} \) with the vertex \( v \). By the conditions for a partial bucket function \( (A_k, f_k) \), \( |\text{segment}(u) - \text{segment}(v)| \leq 1 \), so \( \text{segment}(u) + 1 = \text{segment}(v) \). However, by the conditions for a partial bucket function successor, \( \text{color}(u) > \text{color}(v) \), a contradiction, since consecutive positions of the same color are in distance \( d+1 \).

Let us now limit the number of states. There are at most \( O^{*}(4.383^8) \) partial bucket functions. Integer \( p = O(rd) \) and \( h_p \) keeps position of at most one vertex in each segment, so there are \( O(n^{O(r)}) \) possible pairs \( (H_p, h_p) \). Therefore, in total, we have \( O(4.383^8 n^{O(r)}) \) states. Note that there at most \( n+1 \) successors of a given state, since choosing \( A_2 \setminus A_1 \) defines the successor uniquely. Note that, as checking if a pair \( (A, f) \) is a partial bucket function can be done in polynomial time, checking if a given triple is a state or checking if one state is a successor of the other can be done in polynomial time too.
To obtain the $O(4.383^n n O(r))$-time and space algorithm, we simply seek a path of states as in Lemma 4.9 memoizing visited states. To limit the algorithm to the polynomial space, we do the same trick as in the $O(9.363^n)$ algorithm for BANDWIDTH.

**Lemma 4.10.** Assume that we have states $S_p = (p, (A_p, f_p), (H_p, h_p))$ and $S_q = (q, (A_q, f_q), (H_q, h_q))$ such that $p < q$, $A_p \subseteq A_q$ and $f_p = f_q|_{A_p}$. Let $m = |A_q \setminus A_q|$. Then one can check if there exists a sequence of states $S_i = (i, (A_i, f_i), (H_i, h_i))$ for $i = p, p+1, \ldots, q$ such that the state $S_{i+1}$ is a successor of the state numbered $S_i$ in time $O(4^n n O(r \log m))$.

**Proof.** First, let us consider the case when $m = 1$. We guess the state $A_k$ such that $|A_k| = s$. We need $A_p \subseteq A_k \subseteq A_q$ and $f_k = f_q|_{A_k}$, so we have only roughly $2^m$ possibilities for $(A_k, f_k)$ and $O(d r) = O(n \tilde{E})$ possibilities for the index $k$. As always, there are $O(2^m)$ possible guesses for $(H_k, h_k)$. We recursively check if there is a sequence of states from $S_p$ to $S_k$ and from $S_k$ to $S_q$. Since at each step we divide $m$ by 2, finally we obtain an $O(4^n n O(r \log m))$ time bound.

Again we set $\alpha := 0.5475$.

1. We guess the state $S_k = (k, (A_k, f_k), (H_k, h_k))$ such that $|A_k| = \lfloor \alpha n \rfloor$. By Theorem 2.7 with $N = n$, we can enumerate all partial bucket extensions in $O(4.383^n)$. We enumerate all partial bucket functions, guess $p$ and $(H_k, h_k)$ and drop those combinations that are not states. Note that there are $O(n \tilde{E})$ possible guesses for $(H_k, h_k)$ and $d r \leq n^2$ guesses for $p$.

2. Using Lemma 4.10 check if there is a path of states from $(0, (0, \emptyset), (0, \emptyset))$ to $S_k$. This phase works in time $4^{\alpha n} n O(r \log n)$. In total, for all $(A_k, f_k)$, this phase works in time $O^*(4.383^n \cdot 4^{\alpha n} n O(r \log n)) = O(9.363^n n O(r \log n))$.

3. Guess the final state $S_{r(d+1)} = (r(d+1), (V, f_r(d+1)), (H_r(d+1), h_r(d+1)))$: $f_r(d+1)$ needs to be a bucket extension of the partial bucket function $(A_k, f_k)$. By Lemma 2.6 bucket extensions can be enumerated with polynomial delay. We guess $h_r(d+1)$ and simply drop those guesses that do not form states. By Lemma 2.5 with $N = r$, there will be at most $O^*(5^n)$ pairs of states $(A_k, f_k)$ and $(V, f_r(d+1))$. We have $n \tilde{E}$ possibilities for $h_r(d+1)$.

4. Using Lemma 4.10 check if there is a path from the state $S_k$ to $S_{r(d+1)}$. This phase works in time $4^{(1-\alpha)n} n O(r \log n)$. In total, for all $S_k$ and $S_{r(d+1)}$ this phase works in time $O^*(5^n 4^{(1-\alpha)n} n O(r \log n)) = O(9.363^n n O(r \log n))$.

**Theorem 4.11.** The DISTORTION problem can be solved in $O(4.383^n)$ time and space. If we are restricted to polynomial space, the extended DISTORTION problem can be solved in $O(9.363^n)$ time.

**Proof.** We almost repeat the argument from [9]. First, we may guess the number of nonempty segments needed to embed $G$ into a line with a pushing embedding $\pi$ with distortion at most $d$. Denote this number by $r$, i.e., $r = \lfloor \max \{ \pi(v) : v \in V(G) \}/(d+1) \rfloor$. Note that the original
DISISTRIBUTION problem can be represented as an extended case with $H = G$ and $Z = \bar{\pi} = \emptyset$ and with guessed $r$.

If $r < n / \log^3(n)$, the thesis is straightforward by applying Theorem 4.4. Therefore, let us assume $r \geq n / \log^3(n)$. As every segment from 1 to $r$ contains at least one vertex in a required pushing embedding $\pi$, by simple counting argument, there needs to be a segment $r/4 \leq k \leq 3r/4$ such that there are at most $4n/r \leq 4\log^3(n)$ vertices assigned to segments $k$ and $k+1$ in total by $\pi$. We guess: segment number $k$, vertices assigned to segments $k$ and $k+1$ and values of $\pi$ for these vertices. We discard any guess that already makes some edge between guessed vertices longer than $d$. As $d, r = O(n)$, we have $n^{O(\log^3 n)}$ possible guesses.

Let $Y$ be the set of vertices assigned to segments $k$ and $k+1$ and look at any connected component $C$ of $G[V \setminus Y]$. Note that if $C$ has neighbours in both segment $k$ and $k+1$, the answer is immediately negative. Moreover, as $G$ was connected, $C$ has a neighbour in segment $k$ or $k+1$. Therefore we know, whether vertices from $C$ should be assigned to segments $1, 2, \ldots, k-1$ or $k+2, \ldots, r$. The problem now decomposes into two subproblems: graphs $H_1$ and $H_2$, such that $H_1$ should be embedded into segments $1$ to $k$ and $H_2$ should be embedded into segments $k+1$ to $r$; moreover, we demand that the embeddings meet the guesses values of $\pi$ on $Y$.

The subproblems are in fact instances of extended DISTRIBUTION problem and can be decomposed further in the same fashion until there are at most $n / \log^3(n)$ segments in one instance. The depth of this recurrence is $O(\log r) = O(\log n)$, and each subproblem with at most $n / \log^3(n)$ can be solved by algorithm described in Theorem 4.4. Therefore, finally, we obtain an algorithm that works in $O(4.383^n)$ time and space and an algorithm that works in $O(9.363^n)$ time and polynomial space.

References

[1] Mihai Badoiu, Kedar Dhamdhere, Anupam Gupta, Yuri Rabinovich, Harald R{"a}cke, R. Ravi, and Anastasios Sidiropoulos. Approximation algorithms for low-distortion embeddings into low-dimensional spaces. In Proc. of SODA’05, pages 119–128, 2005.

[2] Hans L. Bodlaender, Michael R. Fellows, and Michael T. Hallett. Beyond NP-completeness for problems of bounded width: Hardness for the W hierarchy (extended abstract). In ACM Symposium on Theory of Computing, pages 449–458, 1994.

[3] M. Cygan and M. Pilipczuk. Even faster exact bandwidth. ACM Transactions on Algorithms (to appear).

[4] M. Cygan and M. Pilipczuk. Faster exact bandwidth. In Proc. WG’08, pages 101–109, 2008.

[5] Marek Cygan and Marcin Pilipczuk. Exact and approximate bandwidth. In Proc. of ICALP’09, pages 304–315, 2009.

[6] Uriel Feige. Coping with the NP-hardness of the graph bandwidth problem. In Proc. SWAT’00, pages 10–19, 2000.

[7] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Elena Losievskaja, Frances A. Rosamond, and Saket Saurabh. Parameterized low-distortion embeddings - graph metrics into lines and trees. CoRR, abs/0804.3028, 2008.

[8] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Elena Losievskaja, Frances A. Rosamond, and Saket Saurabh. Distortion is fixed parameter tractable. In Proc. of ICALP’09, pages 463–474, 2009.

[9] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. An exact algorithm for minimum distortion embedding. In Proc. of WG’09, pages 112–121, 2009.
A Bound on the number of partial bucket functions

In this section we prove Theorem 2.7; namely, that for some constant $c < 4.383$ in a connected, undirected graph $G = (V, E)$ with $|V| = n$ there are at most $O(N \cdot c^n)$ bucket functions, where we are allowed to assign values $\{1, 2, \ldots, N\}$ only. Let $c = 4.383 - \varepsilon$ for some sufficiently small $\varepsilon$. We use $c$ instead of simply constant $4.383$ to hide polynomial factors at the end, i.e., to say $O^*(c^n) = O(4.383^n)$.

Let us start with the following observation.

**Lemma A.1.** Let $G' = (V, E')$ be a graph formed by removing one edge from the graph $G$ in a way that $G'$ is still connected. If $(A, f)$ is a bucket function in $G$, then it is also a bucket function in $G'$.

Therefore we can assume that $G = (V, E)$ is a tree. Take any vertex $v_r$ with degree 1 and make it a root of $G$.

In this proof we limit not the number of partial bucket functions, but the number of prototypes, defined below. It is quite clear that the number of prototypes is larger than the number of partial bucket extensions, and we prove that there are at most $O(Nc^n)$ prototypes. Then we show that one can generate all prototypes in $O^*(Nc^n)$ time and in polynomial space. This proves that all partial bucket extensions can be generated in $O^*(Nc^n)$ time and polynomial space.

**Definition A.2.** Assume we have a fixed subset $B \subseteq V$. A prototype is a pair $(A, f)$, where $A \subseteq V, f : A \cup B \rightarrow \mathbb{Z}$, such that $(A, f|_A)$ is a partial bucket function, and there exists a bucket extension $\bar{f}$ that is an extension of $f$, not only $f|_A$.

**Lemma A.3.** For any fixed $B \subseteq V$ the number of partial bucket functions is not greater than the number of prototypes.

**Proof.** Let us assign to every prototype $(A, f)$ the partial bucket function $(A, f|_A)$. To prove our lemma we need to show that this assignment is surjective. Having a partial bucket function $(A, f)$, take any its bucket extension $\bar{f}$ and look at the pair $(A, f|_{A \cup B})$. This is clearly a prototype, and $(A, f)$ is assigned to it in the aforementioned assignment.

Before we proceed to main estimations, we need a few calculations. Let $\alpha = 4.26, \beta = 3$ and $\gamma = 5.02$.

**Lemma A.4.**

\[
2c^{n-1} + \sum_{k=1}^{\infty} (2k-1)c^{n-k} = c^n\left(\frac{2}{c} + \frac{2c}{(c-1)^2} - \frac{1}{c-1}\right)
\]

**Proof.**

\[
\sum_{k=1}^{\infty} kc^{-k} = \frac{1}{c} \sum_{k=0}^{\infty} (k+1)c^{-k} = \frac{1}{c} \left(\frac{1}{1-x}\right)'\bigg|_{x=\frac{1}{c}} = \frac{c}{(c-1)^2}
\]  \hfill (A.1)
\[ 2c^{n-1} + \sum_{k=1}^{\infty} (2k-1)c^{n-k} = \]
\[ = c^n \left( 2 \sum_{k=1}^{\infty} k c^{-k} - \sum_{k=1}^{\infty} c^{-k} + 2c^{-1} \right) = \]
\[ = c^n \left( 2 \cdot \frac{c}{e} + \frac{2c}{(e-1)^2} - \frac{1}{e-1} \right) \]

**Corollary A.5.** For our choice of values for \(\alpha, \gamma\) and \(c\) we obtain

\[ 2c^{n-1} + \sum_{k=1}^{\infty} (2k-1)c^{n-k} \leq c^n \left( 1 - \max \left( \frac{6}{\alpha c^2}, \frac{15}{\gamma c^3} \right) \right). \]

**Lemma A.6.**

\[ \sum_{k=1}^{\infty} 2k c^{n-k} = c^n \cdot \frac{2c}{(e-1)^2} \]

**Proof.** This is a straightforward corollary from Equation A.1.

**Corollary A.7.** For our choice of values for \(\beta, \gamma\) and \(c\) we obtain

\[ \sum_{k=1}^{\infty} 2k c^{n-k} \leq c^n \left( 1 - \max \left( \frac{7}{\beta c^2}, \frac{13}{\gamma c^3} \right) \right). \]

Let us proceed to the main estimations.

**Lemma A.8.** Let \(G\) be a path of length \(n + 1\) — graph with \(V = \{v_0, v_1, v_2, \ldots, v_n\}, E = \{(v_i, v_{i+1}) : 0 \leq i < n\}\). Let \(B = \{v_0\}\). Fix any \(j \in \mathbb{Z}\). Let \(T(n)\) be the number of prototypes \((A, f)\) satisfying \(v_0 \in A\) and \(f(v_0) = j\). Then \(T(n) \leq \alpha \cdot c^{n-1}\).

**Proof.** Let us denote \(T(x) = 0\) for \(x \leq 0\). This satisfies \(T(x) \leq \alpha c^{x-1}\). We use the induction and start with calculating \(T(1)\) and \(T(2)\) manually.

If \(n = 1\) we have \(f(v_1) \in \{j-1, j, j+1\}\) if \(v_1 \in A\), and one prototype if \(v_1 \notin A\), so \(T(1) = 4 < \alpha\).

If \(n = 2\), we consider several cases. If \(v_1 \in A\) we have \(f(v_1) \in \{j-1, j, j+1\}\) and \(T(1)\) possibilities for \(A \setminus \{v_0\}\) and \(f|_{A \setminus \{v_0\}}\). If \(A = \{v_0, v_2\}\), \(f(v_2) \in \{j-1, j, j+1\}\) due to the conditions for a partial bucket extension \(f\). There is also one state with \(A = \{v_0\}\), ending up with \(T(2) = 3 \cdot 4 + 3 + 1 = 16 < \alpha c\).

Let us recursively count interesting prototypes for \(n \geq 3\). There is exactly one prototype \((A, f)\) with \(A = \{v_0\}\). Otherwise let \(k(A) > 0\) be the smallest positive integer satisfying \(v_{k(A)} \in A\). Let us count the number of prototypes \((A, f)\), such that \(k(A) = k\) for fixed \(k\).

For \(k = 1\) we have \(f(v_1) \in \{j-1, j, j+1\}\) and, having fixed value \(f(v_1)\), we have \(T(n-1)\) ways to choose \(A \setminus \{v_0\}\) and \(f|_{A \setminus \{v_0\}}\).
For $k > 1$ we have $j - k + 1 \leq f(v_k) \leq j + k - 1$, due to the conditions for a partial bucket extension $\bar{f}$, so we have $(2k - 1)$ ways to choose $f(v_k)$ and $T(n - k)$ ways to choose $A \setminus \{v_0, v_1, \ldots, v_{k-1}\}$ and $f_{A \setminus \{v_0, v_1, \ldots, v_{k-1}\}}$ if $k < n$ and 1 way if $k = n$.

Therefore we have for $n \geq 3$:

$$T(n) \leq 1 + 3T(n - 1) + \sum_{k=2}^{n-1} (2k - 1)T(n - k) + 2n - 1 \leq 2n + 2T(n - 1) + \sum_{k=1}^{\infty} (2k - 1)T(n - k).$$

Note that for $n \geq 3$ we have $2n \leq \frac{6}{\alpha c} \cdot \alpha c^{n-1}$, as we have an equality for $n = 3$ and the right side grows significantly faster than the left side for $n \geq 3$. Using Corollary [A.5] we obtain:

$$T(n) \leq \alpha c^{n-1}. \quad \square$$

**Lemma A.9.** Let $G$ be a path of length $n + 1$ — graph with $V = \{v_0, v_1, v_2, \ldots, v_n\}$, $B = \{v_0\}$ and $E = \{(v_i, v_{i+1}) : 0 \leq i < n\}$. Fix any $j \in \mathbb{Z}$. Let $T'(n)$ be the number of prototypes $(A, f)$ satisfying $v_0 \notin A$ and $f(v_0) = j$. Then $T'(n) \leq \beta c^{n-1}$.

**Proof.** Write the formula for $T'$ using previously bounded $T$. We start with calculating $T'(1)$ and $T'(2)$ manually.

If $n = 1$, if $v_1 \in A$ we have $f(v_1) \in \{j, j + 1\}$ and one prototype with $A = \emptyset$, so $T'(1) = 3 \leq \beta$.

If $n = 2$, we have one prototype with $A = \emptyset$, four prototypes if $A = \{v_2\}$ (since then $f(v_2) \in \{j - 1, j, j + 1, j + 2\}$) and $2 \cdot T(1)$ prototypes if $v_1 \in A$ (since $f(v_1) \in \{j, j + 1\}$). Therefore $T'(2) = 1 + 4 + 2 \cdot 4 = 13 < \beta c$.

Let us assume $n \geq 3$.

There is exactly one prototype $(A, f)$ with $A = \emptyset$. Otherwise let $k(A) > 0$ be the smallest positive integer satisfying $v_{k(A)} \in A$. Let us count the number of prototypes $(A, f)$ such that $k(A) = k$ for fixed $k$.

Note that, due to the conditions for a partial bucket extension $\bar{f}$, $j - k + 1 \leq f(v_k) \leq j + k$; there are $2k$ ways to choose $f(v_k)$. There are $T(n - k)$ ways to choose $A \setminus \{v_0, v_1, \ldots, v_{k-1}\}$ and $f_{A \setminus \{v_0, v_1, \ldots, v_{k-1}\}}$ for $k < n$ and 1 way for $k = n$, leading us to inequality

$$T'(n) \leq 1 + 2n + \sum_{k=1}^{\infty} 2kT(n - k).$$

Note that for $n \geq 3$ we have $2n + 1 \leq \frac{7}{\beta c} \cdot \beta c^{n-1}$, as we have equality for $n = 3$ and the right side grows significantly faster than the left side for $n \geq 3$. Therefore, using Corollary [A.7] we obtain

$$T'(n) \leq \beta c^{n-1} \quad \square$$

13
Lemma A.10. Let $G$ be a path of length $n + 1$ — graph with $V = \{v_0, v_1, v_2, \ldots, v_n\}$, $B = \{v_0, v_n\}$ and $E = \{(v_i, v_{i+1}) : 0 \leq i < n\}$. Fix any $j \in \mathbb{Z}$. Let $S(n)$ be the number of prototypes $(A, f)$ satisfying $v_0 \in A$ and $f(v_0) = j$. Then $S(n) \leq \gamma c^{n-1}$. Moreover, at least $0.4S(n)$ of these prototypes $(A, f)$ satisfy $v_n \notin A$.

Proof. As in the estimations of $T(n)$, we use induction and write a recursive formula for $S$. Let $S(x) = 0$ for $x \leq 0$.

We start with calculating $S(1)$, $S(2)$ and $S(3)$ manually. If $n = 1$, if $v_1 \in A$ we have $f(v_1) \in \{j - 1, j, j + 1\}$ and if $v_1 \notin A$ we have $f(v_1) \in \{j - 1, j\}$, thus $S(1) = 5 \leq \gamma$ and $2 = 0.4S(1)$ of these prototypes satisfy $v_1 \notin A$.

If $n = 2$, we consider several cases, as in calculations of $T(2)$. If $v_1 \in A$, we have $f(v_1) \in \{j - 1, j, j + 1\}$ thus $3 \cdot S(1)$ possibilities and out of them $3 \cdot 2$ possibilities satisfy $v_2 \notin A$. If $A = \{v_0, v_2\}$ we have $f(v_2) \in \{j - 1, j, j + 1\}$, $3$ possibilities. If $A = \{v_0\}$ we have $f(v_2) \in \{j - 2, j - 1, j, j + 1\}$, $4$ possibilities. In total, $S(2) = 15 + 3 + 4 = 22 \leq \gamma c$ and $3 \cdot 2 + 4 > 0.4S(2)$ of these prototypes satisfy $v_2 \notin A$.

If $n = 3$, we do similarly. If $v_1 \in A$, we have $f(v_1) \in \{j - 1, j, j + 1\}$ thus $3 \cdot S(2)$ possibilities and out of them $3 \cdot 10$ possibilities satisfy $v_3 \notin A$. If $v_1 \notin A$ but $v_2 \in A$ we have $f(v_2) \in \{j - 1, j, j + 1\}$, $3 \cdot S(1)$ possibilities and out of them $3 \cdot 2$ possibilities satisfy $v_3 \notin A$. If $A = \{v_0, v_3\}$ we have $f(v_3) \in \{j - 2, j - 1, j, j + 1, j + 2\}$, $5$ possibilities. If $A = \{v_0\}$ we have $f(v_3) \in \{j - 3, j - 2, j - 1, j, j + 1, j + 2\}$, $6$ possibilities. In total $S(3) = 3 \cdot 22 + 3 \cdot 5 + 5 + 6 = 92 \leq \gamma c^2$, and $3 \cdot 10 + 3 \cdot 2 + 6 = 42 > 0.4S(3)$ of these prototypes satisfy $v_3 \notin A$.

Let us assume $n \geq 4$. If $A = \{v_0\}$, we have $j - n \leq f(v_n) \leq j + n - 1$, $2n$ possible prototypes and all of them satisfy $v_n \notin A$. Otherwise let $k(A)$ be the smallest positive integer such that $v_{k(A)} \in A$. Let us once again count the number of prototypes $(A, f)$, such that $k(A) = k$ for fixed $k$.

As in the estimate of $T(n)$, we have $3$ possible values for $f(v_k)$ when $k = 1$ and $(2k - 1)$ possible values when $k > 1$. For $k < n$ there are $S(n - k)$ possible ways to choose $A \backslash \{v_0, v_1, \ldots, v_{k-1}\}$ and $f_{A \backslash \{v_0, v_1, \ldots, v_{k-1}\}}$ and $1$ way if $k = n$. Moreover for $k < n$ at least $0.4S(n - k)$ of choices satisfy $v_n \notin A$. Therefore:

$$S(n) = 2n - 1 + 2n + 2S(n - 1) + \sum_{k=1}^{n-1} (2k - 1)S(n - k)$$

And at least

$$2n + 0.4 \left(2S(n - 1) + \sum_{k=1}^{n-1} (2k - 1)S(n - k)\right) \geq 0.4S(n)$$

of these prototypes satisfy $v_n \notin A$. For $n \geq 4$ we have $4n - 1 \leq \frac{15}{\gamma c^3} \cdot \gamma c^{n-1}$, so using Corollary A.5 we obtain:

$$S(n) \leq \gamma c^{n-1}$$

$\square$
Lemma A.11. Let $G$ be a path of length $n + 1$ — graph with $V = \{v_0, v_1, v_2, \ldots, v_n\}$, $B = \{v_0, v_n\}$ and $E = \{(v_i, v_{i+1}) : 0 \leq i < n\}$. Fix any $j \in \mathbb{Z}$. Let $S'(n)$ be the number of prototypes $(A, f)$ satisfying $v_0 \notin A$ and $f(v_0) = j$. Then $S'(n) \leq \gamma c^{n-1}$. Moreover, at least $0.4S'(n)$ of these prototypes $(A, f)$ satisfy $v_n \notin A$.

Proof. Similarly to the estimate of $T'$, we write the formula bounding $S'$ with $S$ and use already proved bounds for $S$. We start with calculating $S'(1)$ and $S'(2)$ manually.

If $n = 1$ we have $f(v_1) \in \{j, j + 1\}$ if $v_1 \in A$ and $f(v_1) \in \{j - 1, j, j + 1\}$ if $v_1 \notin A$, thus $S'(1) = 5 \leq \gamma$ and $3 > 0.4S'(1)$ of these prototypes satisfy $v_1 \notin A$.

If $n = 2$ we consider several cases. If $v_1 \in A$ we have $f(v_1) \in \{j, j + 1\}$, thus $2 \cdot S(1)$ possibilities and out of them $2 \cdot 2$ possibilities satisfy $v_2 \notin A$. If $A = \{v_2\}$ we have $f(v_2) \in \{j - 1, j, j + 1, j + 2\}$, 4 possibilities. If $A = \emptyset$ we have $f(v_2) \in \{j - 2, j - 1, j, j + 1, j + 2\}$, 5 possibilities. In total $S'(2) = 2 \cdot 5 + 4 + 5 = 19 \leq \gamma c$, and $2 \cdot 2 + 5 = 9 > 0.4S(2)$ of these prototypes satisfy $v_2 \notin A$.

Let us assume $n \geq 3$. If $A = \emptyset$, we have $j - n \leq f(v_n) \leq j + n$, $2n + 1$ possible prototypes, all satisfying $v_n \notin A$. Otherwise let $k(A)$ be the smallest positive integer such that $v_k(A) \in A$. Let us once again count number of prototypes $(A, f)$, such that $k(A) = k$ for fixed $k$.

As in the estimate of $T'(n)$, we have $2k$ possible values for $f(v_k)$. For $k < n$ there are $S(n - k)$ possible ways to choose $A \setminus \{v_0, v_1, \ldots, v_{k-1}\}$ and $f_{A \setminus \{v_0, v_1, \ldots, v_{k-1}\}}$ and 1 way if $k = n$. Moreover, for $k < n$ at least $0.4S(n - k)$ of choices satisfy $v_n \notin A$. Therefore:

$$S'(n) \leq 2n + 1 + 2n + \sum_{k=1}^{\infty} 2kS(n - k)$$

and at least

$$2n + 1 + 0.4 \sum_{k=1}^{\infty} 2kS(n - k) \geq 0.4S'(n)$$

of these prototypes satisfy $v_n \notin A$. For $n \geq 3$ we have $4n + 1 \leq \frac{13}{\gamma c} \cdot \gamma c^{n-1}$. Using Corollary A.7 we obtain

$$S'(n) \leq \gamma c^{n-1}$$

Let us proceed to the final lemma in this proof. By $B_0 \subseteq V$ we denote the root $v_r$ and the set of vertices with at least two children in $G$, i.e., vertices of degree at least 3. Recall that $v_r$ has degree 1.

Lemma A.12. Let $v_r$ be the root of an $n$ vertex graph $G = (V, E)$ of degree 1 and let $B = B_0$. Assume that $G$ is not a path. Fix $j \in \mathbb{Z}$. Then both the number of prototypes $(A, f)$ with $f(v_r) = j$, $v_r \in A$ and the number of prototypes $(A, f)$ with $f(v_r) = j$, $v_r \notin A$ are at most $\delta c^{n-2}$, where $\delta = \sqrt{0.66\alpha^2 + 0.4\beta^2}$.

Proof. We prove it by induction over $n = |V|$. Let $v$ be the closest to $v_r$ vertex that belongs to $B_0$ different than $v_r$ ($v$ exists as $G$ is not a path) Let $P$ be the path from $v$ to $v_r$, including $v$ and $v_r$ and let $|P|$ be the number of vertices on $P$. Due to Lemma A.10 and Lemma A.11, there are
at most $\gamma c^{|P|-2}$ ways to choose $(A \cap P, f|_{(A \cup B) \cap P})$, and at least 0.4 of these possibilities satisfy $v \notin A$. Let us now fix one of such choices.

Let $G_1, G_2, \ldots, G_k$ be the connected components of $G$ with removed $P$. Let $V_i$ be the set of vertices of $G_i$, and $B_i = B \cap V_i$. For each $1 \leq i \leq k$, we bound the number of possible choices for $(A \cap V_i, f|_{(A \cup B) \cap V_i})$.

If $B_i = \emptyset$ (equivalently $G_i$ is a path) then one can choose $(A \cap V_i, f|_{(A \cup B) \cap V_i})$ on $T(|V_i|) \leq \alpha c^{|V_i|-1}$ or $T'(|V_i|) \leq \beta c^{|V_i|-1}$ ways, depending on whether $v = v_0 \in A$ or $v = v_0 \notin A$ (we use here Lemma A.8 or Lemma A.9 for Corollary A.13).

Otherwise, we use inductive assumption for $G_i$ with added root $v$. In this case we have at most $\delta c^{|V_i|-1}$ possibilities to choose $(A \cap V_i, f|_{(A \cup B) \cap V_i})$.

Let $\mathcal{B} = \{1 \leq i \leq k : B_i = \emptyset\}$, and $\mathcal{A} = \{1, 2, \ldots, k\} \setminus \mathcal{B}$. If $v \in A$, the number of choices for all graphs $G_i$ is bounded by:

$$\left(\prod_{i \in \mathcal{A}} \delta c_i^{|V_i|-1}\right) \cdot \left(\prod_{i \in \mathcal{B}} \alpha c_i^{|V_i|-1}\right) = \delta^{||\mathcal{A}||} \alpha^{||\mathcal{B}||} c^{n-|P|-k}$$

If $v \notin A$, the number of choices for all graphs $G_i$ is bounded by:

$$\left(\prod_{i \in \mathcal{A}} \delta c_i^{|V_i|-1}\right) \cdot \left(\prod_{i \in \mathcal{B}} \beta c_i^{|V_i|-1}\right) = \delta^{||\mathcal{A}||} \beta^{||\mathcal{B}||} c^{n-|P|-k}$$

Therefore, as $\alpha \geq \beta$, the total number of prototypes for $G$ is bounded by

$$\gamma c^{|P|-2} \delta^{||\mathcal{A}||} c^{n-|P|-k} \left(0.6\alpha^{|\mathcal{B}|} + 0.4\beta^{|\mathcal{B}|}\right) = c^{n-2} \left(\gamma c^{-k} \delta^{||\mathcal{A}||} \left(0.6\alpha^{|\mathcal{B}|} + 0.4\beta^{|\mathcal{B}|}\right)\right)$$

Note that $\delta \gamma \leq c^2$. If $|\mathcal{B}| \leq 1$ we have, using that $k \geq 2$ and $0.6\alpha + 0.4\beta \leq \delta \leq c$:

$$\gamma c^{-k} \delta^{||\mathcal{A}||} \left(0.6\alpha^{|\mathcal{B}|} + 0.4\beta^{|\mathcal{B}|}\right) \leq \gamma c^{-k} \delta^k \leq \delta.$$ 

Otherwise, if $|\mathcal{B}| \geq 2$ we have, as $\beta \leq \alpha \leq c$ and $\delta \leq c$:

$$\gamma c^{-k} \delta^{||\mathcal{A}||} \left(0.6\alpha^{|\mathcal{B}|} + 0.4\beta^{|\mathcal{B}|}\right) \leq \gamma c^{-k} \delta^{||\mathcal{A}||} \left(0.6\alpha^{|\mathcal{B}|} + 0.4\alpha^{|\mathcal{B}|-2} \beta^2\right)$$

$$= \gamma c^{-k} \delta^{||\mathcal{A}||} \alpha^{|\mathcal{B}|-2} \beta^2 \leq \delta.$$ 

Thus the bound is proven.

**Corollary A.13.** The number of all prototypes satisfying $f(v_r) \in \{1, 2, \ldots, N\}$ is at most $N \cdot \max(\alpha, \delta) \cdot c^{n-2} = O(Nc^n)$.

To finish up the proof of theorem 2.7, we need to show the following lemma.

**Lemma A.14.** Fix $B = B_0$. All prototypes can be generated in polynomial space and in $O^*(Nc^n)$ time.
Proof. We assume that \( G = (V, E) \) is a tree rooted at \( v_r \). Otherwise, we may take any spanning tree of \( G \), generate all prototypes for this tree, and finally for each prototype in the spanning tree check if this is a prototype in the original graph \( G \) too.

First we guess \( f(v_r) \) and guess the set \( A \). Then we go in the root–to–leaves order in \( G \) and guess values of \( f \) for vertices in \( A \cup B \). Whenever we encounter a vertex \( v \in A \cup B \) we look at its closest predecessor \( w \in A \cup B \). Let \( d \) be the distance between \( v \) and \( w \). We iterate over all possibilities \( f(v) \in \{ f(w) - d, f(w) - d + 1, \ldots, f(w) + d \} \); however the following options are forbidden due to the conditions for the bucket extension:

- if \( v \in A \), \( w \in A \) and \( d > 1 \) then \( f(v) = f(w) - d \) and \( f(v) = f(w) + d \) are forbidden;
- if \( v \in A \) and \( w \notin A \) then \( f(v) = f(w) - d \) is forbidden;
- if \( v \notin A \) and \( w \in A \) then \( f(v) = f(w) + d \) is forbidden.

Since every branch in our search ends up with a valid prototype, the algorithm takes \( O^*(Nc^n) \) time. In memory, we keep only the recursion stack of the search algorithm, and therefore we use polynomial space.

B Omitted proofs

Proof of Lemma 2.6 We construct all valid bucket extensions by a brute-force search. We start with \( f' = f \) and \( B = A \). At one step we have \( A \subseteq B \subseteq V \), \( f' : B \rightarrow V \) such that \( f'|_A = f \) and there exists a bucket extension \( f \) of \((A, f)\) such that \( f|_B = f' \). We take any \( v \in V \setminus B \) such that there exists a neighbour \( w \) of \( v \) that belongs to \( B \) and try to assign \( f'(v) = f'(w) + \varepsilon \), for each \( \varepsilon \in \{-1, 0, 1\} \). At every step, we use the algorithm from Lemma 2.4 to check the condition if \( f' \) can be extended to a valid bucket extension of \((A, f)\). This check ensures that every branch in our search algorithm ends up with a bucket extension. Therefore we generate all bucket extensions with a polynomial delay and in polynomial space.

Proof of Lemma 4.3 First, assume \( \pi \) has distortion at most \( d \). Then for each \( uv \in E \) we have \( |\pi(u) - \pi(v)| \leq d \). Since segments are of size \( d + 1 \), this implies that \( |\text{segment}(u) - \text{segment}(v)| \leq 1 \). Moreover, the distance between positions of the same color in consecutive segments is exactly \( d + 1 \), which implies that if \( \text{segment}(u) + 1 = \text{segment}(v) \) then \( \text{color}(u) > \text{color}(v) \).

In the other direction, assume that for some \( u, v \in V \) we have \( k = d_G(u, v) |\pi(u) - \pi(v)| > dk \). Let \( u = u_0, u_1, \ldots u_k = v \) be the path of length \( k \) between \( u \) and \( v \). Then, for some \( 0 \leq i < k \) we have \( |\pi(u_{i+1}) - \pi(u_i)| > d \). This implies that \( \text{segment}(u_{i+1}) \neq \text{segment}(u_i) \), w.l.o.g. assume that \( \text{segment}(u_i) + 1 = \text{segment}(u_{i+1}) \). However, since consecutive positions of the same color are in distance \( d + 1 \), this implies that \( \text{color}(u_i) \leq \text{color}(u_{i+1}) \), a contradiction.

17