Shorter Labeling Schemes for Planar Graphs[*]

Marthe Bonamy[†]   Cyril Gavoille[‡]   Michał Pilipczuk[§]

April 20, 2020

Abstract

An *adjacency labeling scheme* for a given class of graphs is an algorithm that for every graph G from the class, assigns bit strings (labels) to vertices of G so that for any two vertices u, v, whether u and v are adjacent can be determined by a fixed procedure that examines only their labels. It is known that planar graphs with n vertices admit a labeling scheme with labels of bit length $(2 + o(1)) \log n$. In this work we improve this bound by designing a labeling scheme with labels of bit length $(\frac{4}{3} + o(1)) \log n$. All the labels of the input graph can be computed in polynomial time, while adjacency can be decided from the labels in constant time.

In graph-theoretical terms, this implies an explicit construction of a graph on $n^{4/3+o(1)}$ vertices that contains all planar graphs on n vertices as induced subgraphs, improving the previous best upper bound of $n^{2+o(1)}$.

Our labeling scheme can be generalized to larger classes of topologically-constrained graphs, for instance to graphs embeddable in any fixed surface or to k-planar graphs for any fixed k, at the cost of larger second-order terms.

Keywords: planar graphs, labeling scheme, universal graphs

[†]CNRS-LaBRI, University of Bordeaux, France, `marthe.bonamy@u-bordeaux.fr`.

[‡]LaBRI, University of Bordeaux, France, `gavoille@labri.fr`.

[§]Institute of Informatics, University of Warsaw, Poland, `michal.pilipczuk@mimuw.edu.pl`.

1 Introduction

When representing graphs, say with adjacency lists or matrices, vertex identifiers usually do not play any particular role with respect to the structure of the graph: they are essentially just pointers in the data structure. In contrast, a graph is *implicitly represented* when each vertex of the graph is associated to more information so that adjacency, for instance, can be efficiently determined from the identifiers without the need of any global data-structure (cf. [KNR88, Spi03]). For example, if G is an interval graph with n vertices, one can associate with each vertex u some interval $I(u) \subseteq [1, 2n]$ with integer endpoints so that u, v are adjacent if and only if $I(u) \cap I(v) \neq \varnothing$. Clearly, no adjacency lists or matrices are required anymore. Although G may have a quadratic number of edges, such an implicit representation uses $2 \log n + O(1)$ bits per vertex[1], regardless of its degree, which is asymptotically optimal [GP08]. Compact representations have several advantages, not only for the memory storage, but also from algorithmic perspectives. For instance, given a succinct representation, BFS traversal can be done in $O(n)$ time [RLDL94, ACJR19], even if the graph has $\Omega(n^2)$ edges. Speedups due to succinct representations are ubiquitous in the design of algorithms and data structures.

Formally introduced by Peleg [Pel00, Pel05], *informative labeling schemes* present a way to formalize implicit representations of graphs. For a given function Π defined on pairs of vertices of a graph from some given class of graphs, an informative labeling scheme has two components: an *encoding algorithm* that associates with each vertex a piece of information (label); and a *decoding algorithm* that computes $\Pi(u, v, G)$, the value of Π applied on vertices u, v of the graph G. The input of the decoding algorithm consists solely of the labels of u and of v, with no other information provided. So, finding an implicit representation of a graph G can be restated as computing an *adjacency labeling scheme* for G, that is, an informative labeling scheme where $\Pi(u, v, G)$ is TRUE if and only if u, v are adjacent in G.

In this paper we will focus on such adjacency labeling schemes (referred to as *labeling schemes* from now on), but many functions Π other than adjacency are of great interest. Among them there are labeling schemes designed for ancestry [FK10b] and lowest common ancestors in rooted trees [AGKR04, AHGL14], distances [GU16, GKU16, AGHP16, FGNW17] and forbidden-set distances [ACGP16], compact routing [FG01, TZ01, RT15], flow problems [KKKP04], and many others. We refer to [GP03], and references therein, for a survey of informative labeling schemes and their applications in distributed computing, and also to [Rot16] for a survey on recent developments in labeling schemes specialized for trees.

Planar graphs. Planar graphs are perhaps one of the most studied class of graphs in this area, due to the wide variety of their implicit representations. To mention just a few, planar graphs are contact graphs of circles [Koe36], of 3D boxes [Tho86], of triangles [dFOdMR94], and more recently, of L-shapes [GIP18]. They also have 1-string representations [CGO10], and their incidence graphs form posets of dimension three [Sch89]. Each of these representations leads to a labelling scheme where each vertex can be encoded using a label consisting of $O(\log n)$ bits, independent of its degree.

The first explicit bound on the label length, given by Kannan et al. [KNR88], was $4 \lceil \log n \rceil$ bits. Using the fact that planar graphs have arboricity at most three together with a labeling scheme for forests with label length $\log n + o(\log n)$, one can achieve also a similar $3 \log n + o(\log n)$ upper bound for planar graphs, where the lower-order term $o(\log n)$ directly depends on the second-order term of the bound for forests. It was a challenging question to optimize this second-order for forests. It has been successively reduced from $O(\log \log n)$ [Chu90] to $O(\log^* n)$ [AR02], and then to a constant only recently by Alstrup et al. [ADBTK17]. As explained above, this leads to an upper bound of $3 \log n + O(1)$ for planar graphs.

---

[1] Throughout the paper, by $\log n$ we denote the binary logarithm of n.

By improving the labeling scheme for bounded treewidth graphs, namely from $O(k \log n)$ [KNR88] to $\log n + O(k \log \log n)$, Gavoille and Labourel [GL07] showed that partitioning the edges of a planar graph into two bounded treewidth subgraphs, rather than into three forests, leads to a shorter representation: with labels consisting of $2 \log n + O(\log \log n)$ bits. Until this work, this has been the best known upper bound for planar graphs.

The best known results for several subclasses of planar graphs are reported in Table 1.

Graph classes (with n vertices)	Upper bound (label length in bits)	References
maximum degree-2	$\log n + O(1)$	[AABTKS16, But09, ELO08]
caterpillars	$\log n + O(1)$	[BGL06]
bounded degree trees	$\log n + O(1)$	[Chu90]
bounded depth trees	$\log n + O(1)$	[FK10a]
trees	$\log n + O(1)$	[ADBTK17]
bounded degree outerplanar	$\log n + O(1)$	[Chu90, AR14]
outerplanar	$\log n + O(\log \log n)$	[GL07]
bounded treewidth planar	$\log n + O(\log \log n)$	[GL07]
maximum degree-4 planar	$\frac{3}{2} \log n + O(\log \log n)$	[AR14]
bounded degree planar	$2 \log n + O(1)$	[Chu90]
planar	$2 \log n + O(\log \log n)$	[GL07]
planar	$\frac{4}{3} \log n + O(\log \log n)$	**[this paper]**

Table 1: State-of-the-art for adjacency labeling schemes on planar graphs and some subclasses. The bounds from references [Chu90, ELO08, But09] come from induced-universal graphs, whereas all the others come from labeling schemes. The only known lower bound for planar graphs is $\log n + \Omega(1)$.

Our contribution. In this work we present a new labeling scheme for planar graphs that uses labels of length bounded[2] by $\frac{4}{3} \log n$. Note that this not only improves the previously best known bound of $2 \log n$ for general planar graphs [GL07], but even the refined bound of $\frac{3}{2} \log n$ for the case of planar graphs of maximum degree 4 [AR14].

The main ingredient of our result is the recent product structure theorem of Dujmović et al. [DJM+19], which says the following: Every planar graph G is a subgraph of a graph of the form $H \boxtimes P$, where H is a graph of treewidth at most 8, P is a path, and $\boxtimes$ denotes the *strong graph product* (see Section 2 for a definition). Moreover, H, P, and a subgraph embedding witnessing this can be found in polynomial time.

The first step in our proof is the design of an auxiliary labeling scheme with labels of length $\log n + \log d$, assuming that the graph G in question is given together with an embedding into $H \boxtimes P$, where H has bounded treewidth and P is a path of length d. This parameterized bound is never worse than the currently best known bound of $2 \log n$, because we may always assume $d < n$, but later in the general case we use it for $d = O(n^{1/3})$. We remark that the proof of the product structure theorem of Dujmović et al. [DJM+19]

---

[2] For brevity, in this informal exposition we ignore additive terms of lower order $o(\log n)$.

in fact yields a subgraph embedding into $H \boxtimes P$ where P has length bounded by the diameter of the considered graph G, so as a side result we obtain a labeling scheme for planar graphs with diameter d that uses labels of length bounded by $\log n + \log d$.

The second step — the main case — relies on the layering technique applied on the structure provided by the product structure theorem. Precisely, for a given planar graph G we compute a subgraph embedding φ of G into $H \boxtimes P$, where H is a graph of bounded treewidth and P is a path (with no nontrivial bound on its length). We choose a parameter $d \geqslant 3$ (which will be set later) and divide $H \boxtimes P$ into blocks of width d; that is, each block is of the form $H \boxtimes Q$ where Q is a subpath of P consisting of d consecutive vertices. By mapping the blocks through φ^{-1} back to G, we thus divide G into *strips*, where each strip can be embedded into $H \boxtimes Q$ where Q has length $d - 1$. These strips are separated by *borders* whose union is a graph on $O(n/d) = O(n^{2/3})$ vertices and of constant treewidth. Using the known bounds for graphs of bounded treewidth [GL07], for this border graph we can compute a labeling λ_1 with labels of length $\log(n/d)$. On the other hand, to the union of strips we can apply the auxiliary labeling scheme explained in the previous paragraph, and thus obtain a labeling λ_2 for the strips with labels of length $\log n + \log d$.

At this point, superposing the two schemes λ_1 and λ_2 would give a labeling scheme of length $2 \log n$. This is because vertices appearing at the borders of strips have to inherit labels from both labelings: $\log(n/d)$ from λ_1 and $\log n + \log d$ from λ_2, which sums up to $2 \log n$. So far, this yields no improvement over the previous results. However, by revisiting the scheme for graphs of bounded treewidth we are able to show that for vertices at the borders — whose number is $O(n/d)$ — the labeling λ_2 can use much shorter labels: only of length $\log(n/d)$ instead of $\log n + \log d$. Hence, the combined labels of border vertices are of length at most $2 \log(n/d)$, implying that every vertex receives a label of length bounded by

$$\max \left\{ \log n + \log d, \ 2 \log(n/d) \right\}.$$

This expression is minimized for $d = n^{1/3}$ and then evaluates to $\frac{4}{3} \log n$, the desired bound.

Finally, we observe that the only property implied by planarity that we used in our labeling scheme is the product structure given by the theorem of Dujmović et al. [DJM+19]. Precisely, if we assume that we work with a class of graph $\mathcal{C}$ such that every graph $G \in \mathcal{C}$ admits a polynomial-time computable subgraph embedding into a graph $H \boxtimes P$, where H has constant treewidth and P is a path, then the whole reasoning goes through. We call such graph classes *efficiently flat* and choose to work throughout the paper with this abstract property alone, instead of the concrete case of planar graphs. The reason for this is that following the result of Dujmović et al. [DJM+19] for planar graphs, many more general classes of graphs have been rendered efficiently flat, for instance graphs embeddable into any fixed surface [DJM+19], or k-planar graphs for any fixed k [DMW19] (see Section 2 for more examples). Consequently, our result gives a labeling scheme of length $\frac{4}{3} \log n$ for all these classes.

In all our labeling schemes, given the input graph we can compute the labeling of its vertices in polynomial time, while the adjacency can be determined from the labels in constant time.

Connections with universal graphs. It has been observed in [KNR88] that the design of labeling schemes with short labels is tightly connected with the construction of small induced-universal graphs. Recall that a graph $\mathcal{U}$ is induced-universal for a given set of graphs $\mathcal{S}$ if every graph $G \in \mathcal{S}$ is isomorphic to some induced subgraph of $\mathcal{U}$. Then graphs from $\mathcal{S}$ admit a labeling scheme with k-bit labels if and only if $\mathcal{S}$ has an induced-universal graph $\mathcal{U}$ with at most 2^k vertices, see [KNR88]. Thus, our labeling scheme provides an explicit construction of an induced-universal graph for n-vertex planar graphs that has $n^{4/3+o(1)}$ vertices, improving upon the previously best known bound of $n^{2+o(1)}$, derived from [GL07].

The search for optimum bounds on the sizes of induced-universal graphs is a well-studied topic, see for example the recent developments for general n-vertex graphs [Alo17, AKTZ15] and for n-vertex trees [ADBTK17]. We refer readers interested in this topic to the recent survey of Alstrup et al. [AKTZ19].

Apart from induced-universal graphs, there is also an alternative definition: *edge-universal graphs*. Here, we say that $\mathcal{U}$ is *edge-universal* for a set of graph $\mathcal{S}$ if every graph from $\mathcal{S}$ is a subgraph of $\mathcal{U}$ (not necessarily induced). As far as edge-universal graphs for n-vertex planar graphs are concerned, there are much more concise constructions than in the induced setting. Babai et al. [BCE$^+$82] gave a construction with $O(n^{3/2})$ edges, while if one restricts the question to n-vertex planar graphs with constant maximum degree, then the number of edges can be reduced even to $O(n)$ [Cap02]. However, in general it is unclear how edge-universal graphs can be turned into induced-universal graphs without a significant explosion in the size, see e.g. the discussion in [Chu90].

Subsequent work. After the publication of the conference version of this work [BGP20], Dujmović et al. [DEJ$^+$20] announced a construction of a labeling scheme of optimum length $(1 + o(1)) \log n$ for any efficiently flat class of graphs. Their proof is also based on the product structure, but is much more involved. Moreover, in the scheme of [DEJ$^+$20], the adjacency tests take time $O(\sqrt{\log n \log \log n})$, as compared to constant time for our scheme.

In the conference version of this work [BGP20] we concentrated on the case of planar graphs and, more generally, graphs embeddable into any fixed surface. Instead of using the product structure abstractly, we relied on a more hands-on combinatorial understanding via BFS layerings and partitions with bounded-treewidth quotient graphs. In particular, in several places we relied on auxiliary properties of the considered classes, like being minor-closed. The version presented here relies on the product structure alone and thus is more general.

Organization. In Section 2 we recall the main definitions and results regarding labeling schemes, tree decompositions, and the product structure theorem of Dujmović et al. [DJM$^+$19]. Then in Section 3 we revisit and strengthen the labeling scheme for graphs of bounded treewidth of Gavoille and Labourel [GL07]. In Section 4 we give an auxiliary scheme for graphs for which the product structure theorem yields an embedding into the strong product of a bounded treewidth graph and a *short* path. This result is then used in Section 5 to treat the general case of graphs from an efficiently flat class. We conclude in Section 6 by discussing some further research directions.

2 Preliminaries

We use standard graph notation. For a graph G, the vertex and edge sets of G are denoted by $V(G)$ and $E(G)$, respectively. For $A \subseteq V(G)$, we write $G[A]$ for the subgraph of G induced by A and $G - A$ for the subgraph of G induced by $V(G) \setminus A$. A *subgraph embedding* of a graph H into a graph G is an injective function $\varphi \colon V(H) \to V(G)$ such that $uv \in E(H)$ entails $\varphi(u)\varphi(v) \in E(G)$.

Labeling schemes. The following definition formalizes the concept of labeling schemes.

Definition 1. *Let $\mathcal{C}$ be a class of graphs. An* adjacency labeling scheme *for $\mathcal{C}$ is a pair $\langle \lambda, \xi \rangle$ of functions such that, for every graph $G \in \mathcal{C}$, it holds:*

- *λ is the* Encoder *that assigns to every vertex u of G a different binary string $\lambda(u, G)$; and*

- ξ is the Decoder *that decides adjacency from the labels taken from* G. *More precisely, for every pair* u, v *of vertices of* G, $\xi(\lambda(u,G), \lambda(v,G))$ *is* TRUE *if and only if* u, v *are adjacent in* G.

The *length of the labeling scheme* $\langle \lambda, \xi \rangle$ *is the function* $\ell \colon \mathbb{N} \to \mathbb{N}$ *that maps every* $n \in \mathbb{N}$ *to the maximum length, expressed in the number of bits, of labels assigned by the Encoder in* n-*vertex graphs from* $\mathcal{C}$.

In the above definition we measure the length only in terms of the vertex count n, but we can extend the definition to incorporate auxiliary graph parameters, like diameter or treewidth, in a natural way. Whenever G is clear from the context, we write $\lambda(u)$ as a shorthand for shorthand for $\lambda(u, G)$.

When speaking about the complexity of Encoder and Decoder, we assume RAM model with machine words of bit length $O(\log n)$ and unit cost arithmetic operations.

Tree decompositions. A *tree decomposition* of a graph G is a pair (T, β), where T is a tree and β maps every node x of T to its *bag* $\beta(x) \subseteq V(G)$ so that: for every edge uv of G there exists a node x satisfying $\{u, v\} \subseteq \beta(x)$, and for every vertex u of G, the set $\{x \in V(T) \colon u \in \beta(x)\}$ induces a non-empty, connected subtree of T. The *width* of (T, β) is $\max_{x \in V(T)} |\beta(x)| - 1$, while the *treewidth* of G is the minimum possible width of a tree decomposition of G.

Flatness. For two graphs G and H, the *strong product* of G and H, denoted $G \boxtimes H$, is the graph on vertex set $V(G) \times V(H)$ where two different vertices (u, v) and (u', v') are adjacent if and only if vertices u and u' are equal or adjacent in G, and vertices v and v' are equal or adjacent in H. The following definition describes the key structural property discovered by Dujmović et al. [DJM$^+$19].

Definition 2. *A class of graph* $\mathcal{C}$ *is* flat *if there exists* $w \in \mathbb{N}$ *such that every graph* $G \in \mathcal{C}$ *is a subgraph of some graph of the form* $H \boxtimes P$, *where* H *has treewidth at most* w *and* P *is a path.*

Note that in the above definition one may assume that $|V(H)| \leqslant |V(G)|$, as one can remove every vertex v of H such that no element of the fiber $\{(v, i) \colon i \in V(P)\}$ participates in the subgraph embedding of G into $H \boxtimes P$. Similarly, we may assume that $|V(P)| \leqslant |V(G)|$.

As proved by Dujmović et al. [DJM$^+$19], planar graphs are flat. However, this property carries over to more general classes of topologically-constrained graphs, as the following classes are flat as well:
- graphs of Euler genus g, for every fixed $g \in \mathbb{N}$ [DJM$^+$19];
- every apex-minor-free class [DJM$^+$19];
- every proper minor-closed class with bounded maximum degree [DJM$^+$19];
- $(0, g, k, p)$-nearly embeddable graphs, for all fixed $g, k, p \in \mathbb{N}$ [DJM$^+$19];
- k-planar graphs, for every fixed k [DMW19].

See also [DMW19] for several other examples of flat classes, and [DHJ$^+$20] for a survey of the area.

In our proofs we will need to assume algorithmic aspects of flatness. Precisely, we shall say that a flat class of graph $\mathcal{C}$ is *efficiently flat* if given $G \in \mathcal{C}$, one can in polynomial time compute a graph H of treewidth at most w, for a constant $w \in \mathbb{N}$, a path P, and a subgraph embedding of G into $H \boxtimes P$. Fortunately, a close inspection of the proofs in [DJM$^+$19] shows that all the abovementioned flat classes are actually efficiently flat, so our results apply to all of them.

Throughout the paper we will focus on proving the following result, from which all the corollaries discussed in Section 1 follow.

Theorem 1. *Every efficiently flat class of graphs admits a labeling scheme of length* $\frac{4}{3} \log n + O(\log \log n)$. *The Encoder runs in polynomial time and the Decoder in constant time.*

3 Bounded Treewidth Graphs

Like the construction of [GL07] for planar graphs, our result relies on the labeling scheme developed for bounded treewidth graphs.

Theorem 2 ([GL07]). *For any fixed $k \in \mathbb{N}$, graphs of treewidth at most k admit a labeling scheme of length $\log n + O(k \log \log n)$. The Encoder runs in $O(n \log n)$ time and the Decoder runs in constant time.*

In later sections we significantly rely on the combinatorics behind the proof of Theorem 2. We will need two ingredients:
  (1) an understanding of how encoding and decoding works in the labeling scheme; and
  (2) a strengthening of the result, where we can assume that a prescribed set of at most q vertices receives shorter labels, namely of length $\log q + O(k \log \log n)$.
These two properties are formally stated as follows.

Theorem 3. *For any fixed $k \in \mathbb{N}$, the class of graphs of treewidth at most k admits a labeling scheme $\langle \lambda, \varphi \rangle$ of length $\log n + O(k \log \log n)$ with the following properties:*
  (P1) From any label a one can extract in time $O(1)$ an identifier $\iota(a)$, so that the Decoder may be implemented as follows: given a label a, one may compute in time $O(k)$ a set $\Gamma(a)$ consisting of at most k identifiers so that $\varphi(a, b)$ is TRUE if and only if $\iota(a) \in \Gamma(b)$ or $\iota(b) \in \Gamma(a)$.
  (P2) If the input graph G is given together with a vertex subset Q, then the scheme can assign to the vertices of Q labels of length $\log |Q| + O(k \log \log n)$.
The Encoder works in time $O(n \log n)$ while the Decoder works in constant time.

The proof of Theorem 3 largely follows the approach of Gavoille and Labourel [GL07]. In particular, their scheme achieves property (P1) without any modifications. However, to achieve property (P2) we need to replace a crucial combinatorial element of the proof with a new argument.

The remainder of this section is devoted to the presentation of the proof of Theorem 3, which largely follows the approach of Gavoille and Labourel [GL07]. In Section 3.1 we recall this approach and explain that property (P1) follows from it without any modifications. In Section 3.2 we replace a crucial ingredient of [GL07] with a new argument in order to achieve property (P2) as well.

3.1 Encoding and decoding

We start with a brief presentation of the approach of Gavoille and Labourel [GL07]. Our presentation is a bit simplified compared to that of [GL07], because we choose not to optimize the label length as much as there (e.g. Gavoille and Labourel actually provide an upper bound of $\log n + O(k \log \log (n/k))$ instead of $\log n + O(k \log \log n)$ by a more precise analysis).

First, since the input graph G has treewidth at most k, one can obtain a chordal supergraph G^+ of G on the same vertex set such that G^+ also has treewidth at most k. This can be done as follows: take a tree decomposition of G of width at most k and turn every bag into a clique. Since for fixed k such a tree decomposition can be computed in linear time [Bod96], G^+ can be computed in linear time.

Next, it is well-known that since G^+ is chordal and of treewidth at most k, in linear time we can compute an orientation $\vec{G}$ of G^+ such that every vertex u has at most k out-neighbors in $\vec{G}$, and moreover u together with those out-neighbors form a clique in G^+. For every $u \in V(G)$, let K_u be the set consisting of u and its out-neighbors in $\vec{G}$.

The key idea of the approach of Gavoille and Labourel is to compute a *bidecomposition* of the graph G^+, which is a notion roughly resembling tree decompositions, but actually quite different.

Definition 3. *A* bidecomposition *of a graph H is a pair (T, α), where T is a binary rooted tree and α maps vertices H to nodes of T, so that for every edge uv of H, $\alpha(u)$ and $\alpha(v)$ are related.*

As proved in [GL07], graphs of bounded treewidth admit bidecompositions with small parts. This is the key combinatorial ingredient of the proof.

Lemma 4 (cf. Lemma 1 in [GL07]). *Let G be an n-vertex graph of treewidth at most k. Then there exists a bidecomposition (T, α) of G satisfying the following:*
(A1) $|\alpha^{-1}(x)| = O(k \log n)$ for every node x of T; and
(A2) T has depth at most $\log n$.
Moreover, for every fixed k, given G such a bidecomposition can be constructed in time $O(n \log n)$.

We apply Lemma 4 to the graph G^+, thus getting a suitable bidecomposition (T, α). Based on this, a labeling is constructed as follows.

Consider any $u \in V(G)$. Since K_u is a clique in G^+, it follows that nodes $\{\alpha(v)\}_{v \in K_u}$ are pairwise related. Hence, there exists a path P_u in T starting at the root that contains all nodes $\alpha(v)$ for $v \in K_u$. The second endpoint of P_u is the deepest among nodes $\{\alpha(v)\}_{v \in K_u}$. Let P'_u be the prefix of P_u from the root of T to $\alpha(u)$.

For each node x of T fix an arbitrary enumeration of $\alpha^{-1}(x)$ using index taken from $[0, |\alpha^{-1}(x)|)$. Now, the identifier of vertex u consists of the following pieces of information:

1. The encoding of the path P'_u as a bit string of length $|V(P'_u)| - 1$ that encodes, for consecutive non-root vertices of P'_u, whether they are left or right children.
2. The index of u within $\alpha^{-1}(\alpha(u))$.
3. The depth of $\alpha(u)$ in T.

Since T has depth at most $\log n$ and $|\alpha^{-1}(x)| = O(k \log n)$ for every node x of T, we conclude that the identifier has total length $\log n + \log k + O(\log \log n)$. In addition to the identifier, the label of u contains the following pieces of information:

1. Encoding of the suffix of P_u that is not contained in P'_u; this, together with the information from the identifier, adds up to the encoding of P_u.
2. For every $v \in K_u \setminus \{u\}$, the depth of $\alpha(v)$ in T, the index of v within $\alpha^{-1}(\alpha(v))$, and whether the edge uv belongs to $E(G)$ (it may belong to $E(G^+) \setminus E(G)$).

As shown in [GL07], the above information, together with the identifier, can be encoded in $\log n + O(k \log \log n)$ bits, resulting in the promised upper bound on the label length. Moreover, given the bidecomposition (T, α) the labeling can be computed in linear time, assuming k is fixed.

It is now straightforward to see that from the label of u one can derive the identifiers of the out-neighbors of u in G^+. Indeed, for every $v \in K_u \setminus \{u\}$ the depth of $\alpha(v)$ and the index of v in $\alpha^{-1}(\alpha(v))$ are directly stored in the label of u, while the encoding of the path P'_v can be obtained by taking the encoding of P_u and trimming it to the prefix of length equal to the depth of $\alpha(v)$. With every such out-neighbor v we have also stored the information whether the edge uv is contained in G, or it was added when modifying G to G^+. Hence, given the label $\lambda(u)$ we can compute a set of at most k identifiers of neighbors of u, which is a suitable set $\Gamma(\lambda(u))$. This proves property (P1).

3.2 Saving on labels of a small set of vertices

We now explain how the general approach of Gavoille and Labourel [GL07], presented in the previous section, can be amended to achieve property (P2) as well. The difference is that we replace the usage of Lemma 4 with the following Lemma 5.

Lemma 5. *Let G be an n-vertex graph of treewidth at most k and $S \subseteq V(G)$. Then there exists a bidecomposition (T, α) of G satisfying the following:*
 (B1) $|\alpha^{-1}(x)| = O(k \log n)$ *for every node x of T;*
 (B2) T *has depth at most* $\log n + O(1)$*; and*
 (B3) *for every* $u \in S$, $\alpha(u)$ *is at depth at most* $\log |S| + O(1)$ *in T.*
Moreover, for every fixed k, given G and S such a decomposition can be constructed in time $O(n \log n)$.

Consider the set Q of prescribed vertices as in property (P2), and apply Lemma 5 to G^+ with

$$S = \bigcup_{u \in Q} K_u.$$

We have $|S| \leqslant (k+1) \cdot |Q|$. Hence, in the notation of the previous section, for every $u \in Q$ we have that P_u has at most $\log |S| + O(1) = \log |Q| + O(\log k)$ nodes, while for every other vertex u we have that P_u has at most $\log n + O(1)$ nodes. Plugging this into the analysis of the previous section gives the desired bounds on the lengths of labels in the constructed labeling. Note that thus, property (P1) still holds, while property (P2) is achieved.

We are left with proving Lemma 5. We would like to stress that this is *not* a simple modification of the proof of Lemma 4 presented in [GL07]. The general idea is to recursively decompose the graph, where at each step we use a separator of size $O(k \log n)$ to split the graph into two parts, each containing (roughly) at most half of the remaining vertices and at most half of the remaining vertices of S. In [GL07] only the first objective — halving the total number of vertices — was necessary, and this was relatively easy to achieve using a separator of size $O(k \log n)$. However, the strategy used in [GL07] does not generalize to achieving both objectives at the same time. Hence, our splitting step is based on a different argument.

PROOF (OF LEMMA 5). We first focus on proving the existential statement. Then, at the end, we discuss the algorithmic aspects of the proof.

For a graph H and a nonnegative weight function $\omega \colon V(H) \to \mathbb{R}_{\geqslant 0}$, we write $\omega(H) = \sum_{u \in V(H)} \omega(u)$. We first need a robust understanding of balanced separators in graphs of bounded treewidth, which boils down to a understanding balanced separators in trees. We will use the following well-known claim.

Claim 1. *Let $\varepsilon > 0$, let T be a tree, and let $\omega \colon V(T) \to \mathbb{R}_{\geqslant 0}$ be a nonnegative weight function on the nodes of T. Then there exists a set of nodes $S \subseteq V(T)$ of size at most ε^{-1} such that for every connected component C of $T - S$, we have $\omega(C) \leqslant \varepsilon \cdot \omega(T)$.*

PROOF. The claim is trivial if $\omega(T) = 0$, so assume that $\omega(T) > 0$. Root T in any node r; this induces a natural ancestor/descendant relation. Starting with $S = \varnothing$ and all vertices of S unmarked, we apply the following iterative procedure. As long as the total weight of unmarked vertices is at least $\varepsilon \cdot \omega(T)$, find the deepest node x such that the total weight of unmarked descendants of x (including x itself) is at least $\varepsilon \cdot \omega(T)$, add x to S, and mark x together with all its descendants.

As with every node added to S we mark nodes of total weight at least $\varepsilon \cdot \omega(T)$, the final set S has size at most ε^{-1}. The fact that every connected component of $T - S$ has weight at most $\varepsilon \cdot \omega(T)$ follows easily from always choosing x as the deepest node with the considered property. ⌟

Claim 1 can be generalized to graphs of bounded treewidth as follows.

Claim 2. *Let $\varepsilon > 0$, let H be a graph of treewidth at most k, and let $\omega \colon V(H) \to \mathbb{R}_{\geqslant 0}$ be a nonnegative weight function on the vertices of H. Then there exists a set $Z \subseteq V(H)$ of size at most $\varepsilon^{-1} \cdot (k+1)$ such that for every connected component C of $H - Z$ we have $\omega(C) \leqslant \varepsilon \cdot \omega(H)$.*

PROOF. Let (T, β) be a tree decomposition of H of width at most k. Let us root T in any node r; this induces a natural parent/child relation in T. For a node x of T, we define the *margin* of x as

$$\mu(x) = \beta(x) \setminus \beta(y),$$

where y is the parent of x in T. For the root r we set $\mu(r) = \beta(r)$. For a node $x \in T$, we define

$$\omega'(x) = \omega(\mu(x)).$$

Thus, $\omega' \colon V(T) \to \mathbb{R}_{\geqslant 0}$ is a nonnegative weight function defined on the nodes of T. By observing that $\{\mu(x) \colon x \in V(T)\}$ is a partition of $V(H)$, we see that $\omega'(T) = \omega(H)$.

By applying Claim 1 to tree T with weight function ω', we obtain a suitable node subset $S \subseteq V(T)$ of size at most ε^{-1}. Let $Z = \bigcup_{x \in S} \beta(x)$; then $|Z| \leqslant \varepsilon^{-1} \cdot (k+1)$. To see that each connected component C of $H - Z$ satisfies $\omega(C) \leqslant \varepsilon \cdot \omega(H)$, note that such a component must be entirely contained in $\bigcup_{x \in V(D)} \mu(x)$ for some connected component D of $T - S$, while $\omega'(D) \leqslant \varepsilon \cdot \omega'(T) = \varepsilon \cdot \omega(H)$ by Claim 1. ⌟

We now use Claim 2 to find separators that are suited for constructing a bidecomposition. The idea is that after removing the separator, we need to be able to group the remaining components into two parts that are roughly balanced: both in terms of the number of vertices and in terms of the number of vertices of Q. We prefer to put this condition in an abstract way, using two weight functions. Precisely, we will use the following auxiliary claim.

Claim 3. *Let Ω be a finite set and $\omega_1, \omega_2 \colon \Omega \to \mathbb{R}_{\geqslant 0}$ be two nonnegative weight functions on Ω such that for every element $x \in \Omega$, we have $\omega_1(x) \leqslant \varepsilon \cdot \omega_1(\Omega)$ and $\omega_2(x) \leqslant \varepsilon \cdot \omega_2(\Omega)$, for some $\varepsilon > 0$. Then there exists a partition of Ω into Y and Z such that $\omega_t(W) \leqslant (1/2 + 3\varepsilon) \cdot \omega_t(\Omega)$, for all $W \in \{Y, Z\}$ and $t \in \{1, 2\}$.*

PROOF. If $\omega_1(\Omega) = 0$ then the claim follows easily by packing elements into Y greedily until $\omega_2(Y) \geqslant \omega_2(\Omega)/2$, and defining $Z = \Omega \setminus Y$. A symmetric reasoning can be applied when $\omega_2(\Omega) = 0$. Therefore, we are left with the case when $\omega_1(\Omega) > 0$ and $\omega_2(\Omega) > 0$, so by rescaling the weights we may assume that $\omega_1(\Omega) = \omega_2(\Omega) = 1$.

For $x \in \Omega$, let

$$\xi(x) = \omega_1(x) - \omega_2(x).$$

Note that

$$\sum_{x \in \Omega} \xi(x) = 0 \qquad \text{and} \qquad |\xi(x)| \leqslant 2\varepsilon \quad \text{for every } x \in \Omega.$$

We now inductively define an ordering $x_1, \ldots, x_n$ of the elements of Ω as follows. Assuming $x_1, \ldots, x_{i-1}$ has already been defined, we select:

- x_i to be an arbitrary non-positive element of $\Omega \setminus \{x_1, \ldots, x_{i-1}\}$, in case $\sum_{j=0}^{i-1} \xi(x_j) \geqslant 0$; and

- x_i to be an arbitrary positive element of $\Omega \setminus \{x_1, \ldots, x_{i-1}\}$, in case $\sum_{j=0}^{i-1} \xi(x_j) < 0$.

The existence of an element x_i as above is implied by the fact that $\sum_{x \in \Omega} \xi(x) = 0$. As $|\xi(x)| \leqslant 2\varepsilon$ for every $x \in \Omega$, the following assertion can be proved by a straightforward induction:

$$\left| \sum_{j=0}^{i} \xi(x_j) \right| \leqslant 2\varepsilon \quad \text{for every } i \in \{0, 1, \ldots, n\}. \tag{1}$$

9

Let $p \in \{1, \ldots, n\}$ be the smallest index such that $\sum_{j=0}^{p} \omega_1(x_i) \geqslant \frac{1}{2}$. As $\omega_1(x_p) \leqslant \varepsilon$, we also have $\sum_{j=0}^{p} \omega_1(x_i) \leqslant \frac{1}{2} + \varepsilon$. Thus, if we define

$$Y := \{x_1, \ldots, x_p\} \qquad \text{and} \qquad Z := \{x_{p+1}, \ldots, x_n\},$$

then we have $1/2 \leqslant \omega_1(Y) \leqslant 1/2 + \varepsilon$, implying $\omega_1(Z) \leqslant 1/2$. Further, from (1) we conclude that

$$2\varepsilon \geqslant \left| \sum_{j=0}^{p} \xi(x_j) \right| = |\omega_1(Y) - \omega_2(Y)|,$$

so $1/2 - 2\varepsilon \leqslant \omega_2(Y) \leqslant 1/2 + 3\varepsilon$. As $Z = \Omega \setminus Y$, the left inequality implies that $\omega_2(Z) \leqslant 1/2 + 2\varepsilon$, so all the required properties are fulfilled. ⌟

The next statement applies Claim 3 in the context of a single step of constructing a bidecomposition.

Claim 4. *Let $\varepsilon > 0$, let H be a graph of treewidth at most k, and let $\omega_1, \omega_2 \colon V(H) \to \mathbb{R}_{\geqslant 0}$ be two weight functions on the vertices of H. Then there exists a partition (A, X, B) of $V(G)$ with the following properties:*
- *there is no edge with one endpoint in A and second in B;*
- *$|X| \leqslant \varepsilon^{-1} \cdot (k+1)$; and*
- *$\omega_t(A) \leqslant (1/2 + \varepsilon) \cdot \omega_t(H)$ and $\omega_t(B) \leqslant (1/2 + \varepsilon) \cdot \omega_t(H)$, for $t \in \{1, 2\}$.*

PROOF. Apply Claim 2 to H with weight functions $\omega_1(\cdot)$ and $\omega_2(\cdot)$, yielding suitable separators $Z_1, Z_2 \subseteq V(H)$. Let $X = Z_1 \cup Z_2$. Then $|X| \leqslant \varepsilon^{-1} \cdot (k+1)$ and

$$\omega_t(C) \leqslant \varepsilon \cdot \omega_t(H) \quad \text{for each } t \in \{1, 2\} \text{ and every connected component } C \text{ of } H - X.$$

It therefore suffices to prove that the connected components of $H - X$ can be partitioned into two groups A and B so that the third condition is satisfied.

Let Ω be the set of connected components of $H - X$. By applying Claim 3 to Ω with weight functions ω_1, ω_2 (treated as weight functions on Ω), we find a partition of Ω into Y and Z such that

$$\omega_t(W) \leqslant (1/2 + 3\varepsilon) \cdot \omega_t(\Omega) \leqslant (1/2 + 3\varepsilon) \cdot \omega_t(H),$$

for all $W \in \{Y, Z\}$ and $t \in \{1, 2\}$. It now remains to define $A = \bigcup_{C \in Y} V(C)$ and $B = \bigcup_{C \in Z} V(C)$, and rescale ε by a multiplicative factor of 3 throughout the proof. ⌟

We are finally ready to build the requested bidecomposition. Let $n = |V(G)|$ and fix $\varepsilon = 1/\log n$. W.l.o.g. assume that $\varepsilon \leqslant 1/8$, for otherwise G is of constant size. Define weight functions on vertices of G as follows: $\omega_1(u) = 1$ for each vertex u, and $\omega_2(u) = 1$ for each $u \in S$ and $\omega_2(u) = 0$ for each $u \notin S$. The bidecomposition is constructed using the following recursive procedure which for $R \subseteq V(G)$ constructs a bidecomposition of $G[R]$; we apply it initially to $R = V(G)$.

1. If R is empty, terminate and return an empty bidecomposition.
2. Otherwise, apply Claim 4 to $H = G[R]$ with weight functions $\omega_1(\cdot)$ and $\omega_2(\cdot)$ restricted to R. This yields a partition (A, X, B) of R. Apply the procedure recursively to A and to B in place of R, yielding bidecompositions of $G[A]$ and $G[B]$, respectively. Return a bidecomposition of $G[R]$ obtained by creating a root r, mapping all vertices of X to r, and attaching the bidecompositions of $G[A]$ and $G[B]$ as children of r.

Let (T, α) be the bidecomposition of G obtained in this manner. As $\varepsilon = 1/\log n$, property (B1) is clear from the construction. Therefore, we focus on proving properties (B2) and (B3).

Consider any root-to-leaf path in T and let

$$V(G) = R_0 \supseteq R_1 \supseteq R_2 \supseteq \cdots \supseteq R_d$$

be the consecutive sets R considered by the procedure constructing (T, α) along this path. By the construction we have

$$|R_i| \leqslant (1/2 + \varepsilon)^i \cdot n \qquad \text{for each } i \in \{0, \ldots, d\}.$$

Since the procedure stops when R becomes empty, we have that $|R_d| \geqslant 1$, implying that

$$(1/2 + \varepsilon)^d \cdot n \geqslant 1$$

or equivalently

$$d \leqslant -\log_{1/2+\varepsilon} n = \frac{\log n}{1 + \log\left(\frac{1}{1+2\varepsilon}\right)}.$$

Now observe that

$$\log\left(\frac{1}{1 + 2\varepsilon}\right) = \log\left(1 - \frac{2\varepsilon}{1 + 2\varepsilon}\right) \geqslant -\frac{4\varepsilon}{1 + 2\varepsilon} \geqslant -4\varepsilon.$$

Here, the first inequality follows from $\log(1 - x) \geqslant -2x$ for $x \in [0, 1/2]$, which in turn follows from the concavity of function $t \mapsto \log t$. Therefore, we conclude that

$$d \leqslant \frac{\log n}{1 - 4\varepsilon} \leqslant \log n \cdot (1 + 8\varepsilon) = \log n + 8,$$

where the second inequality follows from $1 \leqslant (1 - 4\varepsilon)(1 + 8\varepsilon)$ being true for $\varepsilon \leqslant 1/8$. We conclude that the height of T is at most $\log n + 8$, so property (B2) is verified.

The proof of property (B3) is analogous: instead of any root-to-leaf path in T, we consider any path from the root to a node x satisfying $\alpha^{-1}(x) \cap S \neq \varnothing$. This concludes the proof of the existential statement.

We now discuss the algorithmic aspects. In the following, k is considered a fixed constant.

It is easy to implement the procedure described in the proof of Claim 1 in linear time by processing the tree T in a bottom-up manner: one keeps track of the total weight of unmarked descendants and when this count reaches $\varepsilon \cdot \omega(T)$, the current node is added to S and the count is zeroed. The proof of Claim 2 essentially boils down to applying Claim 1 to a tree decomposition of width k, with a weight function that can be computed in linear time directly from the definition. As for a fixed k, an optimum-width tree decomposition of a graph of treewidth at most k can be found in linear time [Bod96], this gives a linear time algorithm for finding the set Z promised by Claim 2.

The procedure described in the proof of Claim 3 can be easily implemented in time linear in $|\Omega|$. By combining this with the discussion of Claim 2 from the previous paragraph, for Claim 4 we obtain an algorithm that runs in linear time.

Finally, the construction of the final bidecomposition amounts to applying (the algorithm of) Claim 4 recursively. It can be easily seen that graphs considered in the subcalls at every level of the recursion are disjoint, so the total amount of work used in those subcalls is linear in the size of the input graph. Since the recursion depth is bounded by $\log n$, we conclude that the algorithm runs in time $O(n \log n)$. $\qquad\square$

4 Case of a short path

We now move to the first step of the proof of Theorem 1. Hence, from now on we fix an efficiently flat class $\mathcal{C}$, and we let $w \in \mathbb{N}$ be the constant given by the efficient flatness of $\mathcal{C}$. In the following we treat w as a constant, hence all the constants hidden in the $O(\cdot)$-notation may depend on w.

We now use our understanding of schemes for graphs of bounded treewidth in order to lift it to graphs from $\mathcal{C}$ that can be embedded into $H \boxtimes P$, where H has bounded treewidth and P is a relatively short path. This intermediate result will be exploited in the next section in the general labeling scheme for $\mathcal{C}$.

Lemma 6. *Graphs from $\mathcal{C}$ admit a labeling scheme of length $\log n + \log d + O(\log \log n)$, where we assume that the Encoder is given a graph $G \in \mathcal{C}$ together with a subgraph embedding of G into a graph $H \boxtimes P$, where H has treewidth at most w and P is a path of length d. The Encoder runs in polynomial time and the Decoder in constant time.*

Moreover, if the graph G is provided together with a vertex subset Q, then the Encoder may assign to the vertices of Q labels of length at most $\log |Q| + \log d + O(\log \log n)$.

PROOF. We first focus on proving the initial statement, without the additional vertex subset Q. At the end we shall argue how the refined statement can be obtained using property (P2) of Theorem 3.

Let $G \in \mathcal{C}$ be the input graph, where G has n vertices. We assume that we are also given a subgraph embedding φ of G into $H \boxtimes P$, where H has treewidth at most w and P has length d. As argued, by removing vertices not participating in the image of G under φ, we may assume that $|V(H)| \leqslant n$ and $|V(P)| = d + 1 \leqslant n$. We identify the vertices of P with numbers $\{0, 1, \ldots, d\}$.

Since H has treewidth $w = O(1)$, we may apply Theorem 3 to H. Thus, in polynomial time we can compute a labeling $\kappa(\cdot)$ defined on vertices of H with labels of length $\log n + O(\log \log n)$, for which we have a Decoder working in constant time.

Now, we define a labeling $\lambda(\cdot)$ of G as follows. Take any $u \in V(G)$ and let $(v, i) = \varphi(u)$, where $v \in V(H)$ and $i \in \{0, 1, \ldots, d\}$. Then the label $\lambda(u)$ consists of:

- the label $\kappa(v)$;
- the number i, written in binary;
- a $3(w + 1)$-bit *adjacency code*, which we define in a moment.

The first two pieces of information above are of variable length, so we add to the label a prefix of (fixed) length $2 \log \log n$ that encodes their lengths, so that they can be extracted from the label in constant time. Clearly, the total length of any label constructed in this way is bounded by $\log n + \log d + O(\log \log n)$.

It remains to describe the adjacency code and how the decoding is going to be performed based on it. Recall that, by property (P1), every vertex of v of H is assigned an identifier $\iota(\kappa(v))$ so that from $\kappa(v)$ one can compute a set $\Gamma(\kappa(v))$ of at most w identifiers with the following property: v and v' are adjacent in H if and only if $\iota(\kappa(v)) \in \Gamma(\kappa(v'))$ or $\iota(\kappa(v)) \in \Gamma(\kappa(v'))$. By ordering identifiers lexicographically, we may assume that sets returned by $\Gamma(\cdot)$ are organized as lists[3]. Observe that two vertices u and u' of G may be adjacent only if the following two assertions hold: denoting $\varphi(u) = (v, i)$ and $\varphi(u') = (v', i')$, we must have that

- v and v' are equal or adjacent in H; and
- $|i - i'| \leqslant 1$.

Hence, the adjacency code assigned to a vertex u of G stores the following information: denoting $(v, i) = \varphi(u)$, for each $v' \in \{v\} \cup \Gamma(\kappa(v))$ and $t \in \{-1, 0, 1\}$, we record whether u is adjacent to the unique

---

[3] In the original scheme of [GL07], $\Gamma(\cdot)$ sets are organized into dictionary so that membership can be tested in constant time, independently of the size of $\Gamma(\cdot)$. This refinement does not matter here since the size is bounded by $w = O(1)$.

vertex u' with $\varphi(u') = (v', i + t)$, provided it exists. Note that there is at most one u' as above, because φ is injective.

Given the above description, the decoding can be performed as follows. Suppose we are given labels $\lambda(u)$ and $\lambda(u')$ of two vertices $u, u' \in V(G)$. Denoting $(v, i) = \varphi(u)$ and $(v', i') = \varphi(u')$, from these labels we may consecutively compute:

- numbers i and i';
- labels $\kappa(v)$ and $\kappa(v')$;
- lists $\Gamma(\kappa(v))$ and $\Gamma(\kappa(v'))$; and
- identifiers $\iota(\kappa(v))$ and $\iota(\kappa(v'))$.

Next, we check whether $\iota(\kappa(v)) = \iota(\kappa(v'))$, or $\iota(\kappa(v)) \in \Gamma(\kappa(v'))$, or $\iota(\kappa(v')) \in \Gamma(\kappa(v))$. If this is not the case, then u and u' are not adjacent in G, because v and v' neither equal nor adjacent in H. Otherwise, we check whether $i - i' \in \{-1, 0, 1\}$. Again, if this is not the case, then u and u' are not adjacent in G, because i and i' are neither equal nor adjacent in P. Otherwise, whether u and u' are adjacent can be read from the adjacency code of $\lambda(u)$ or of $\lambda(u')$, depending on which identifier belongs to which list.

From the above description it is clear that the Encoder for this labeling scheme runs in polynomial time, while the Decoder runs in constant time. This concludes the proof of the initial statement, without the additional vertex subset Q. For the additional statement, we simply apply the following modification: we use property (P2) of Theorem 3 to ensure that in the labeling $\kappa(\cdot)$, the vertices of H that appear on the first coordinates of $\varphi(Q)$ receive labels of length $\log |Q| + O(\log \log n)$. Thus, in $\lambda(\cdot)$ the vertices of Q receive labels of total length at most $\log |Q| + \log d + O(\log \log n)$. $\qquad \square$

Remark 1. In the labeling scheme of Lemma 6, we reserve $\lceil \log(d + 1) \rceil$ bits in the label of each vertex u to store the P-*coordinate* of $\varphi(u)$, that is, the number i where $\varphi(u) = (v, i)$. Observe that we may modify the scheme so that for vertices the P-coordinate i is either 0 or d, this piece of information takes $O(1)$ bits. Namely, using 3 first bits we store i is equal to 0, 1, $d - 1$, d, or lies between 2 and $d - 2$. Then the value of i is recorded using $\lceil \log(d + 1) \rceil$ additional bits only when it is between 1 and $d - 1$. It is easy to see that using this way of storing the P-coordinates, the Decoder can verify whether two given P-coordinates (decoded from the labels) differ by at most 1 (and then what is their difference), even when the Decoder does not know the value of d in advance. We will use this optimization in the next section.

Remark 2. The proof of flatness of planar graphs given by Dujmović et al. [DEJ$^+$19] actually provides a subgraph embedding of every planar graph G into a graph of the form $H \boxtimes P$, where H is a graph of treewidth at most 8 and P is a path whose length is bounded by the diameter of G. Hence, from Lemma 6 we can infer that planar graphs admit a labeling scheme of length $\log n + \log d + O(\log \log n)$, where d is the diameter of the input graph.

5   General case

Finally, we are ready to prove our main result, Theorem 1.

PROOF (OF THEOREM 1). Let $G = (V, E) \in \mathcal{C}$ be the input graph on n vertices. Let

$$d = \left\lceil n^{1/3} \right\rceil.$$

W.l.o.g. we assume that $d \geqslant 3$ (or $n > 8$).

By efficient flatness of $\mathcal{C}$, we may compute in polynomial time a graph H of treewidth at most w, a path P, and a subgraph embedding φ of G into $H \boxtimes P$. As before, by removing unnecessary vertices we

may assume that $|V(H)| \leqslant n$ and $|V(P)| \leqslant n$. Letting ℓ be the length of P, we again identify the vertices of P with numbers $\{0, 1, \ldots, \ell\}$.

For $i \in \{0, 1, \ldots, \ell\}$ let

$$L_i = \varphi^{-1}(\{(v, i) \colon v \in V(H)\}),$$

and for $a \in \{0, \ldots, d-1\}$ let

$$W_a = \bigcup_{i \in \mathbb{N} \colon i \equiv a \bmod d} L_i.$$

Note that $\{L_0, L_1, \ldots, L_\ell\}$ and $\{W_0, W_1, \ldots, W_{d-1}\}$ are partitions of V.

When speaking about sets W_a, we consider indices modulo d. Then one of the sets $W_a \cup W_{a+1}$ is small in the following sense.

Claim 5. *There exists $a \in \{0, \ldots, d-1\}$ such that $|W_a \cup W_{a+1}| \leqslant 2n^{2/3}$.*

PROOF. Observe that $\sum_{a \in [0,d)} |W_a \cup W_{a+1}| = 2n$ because every vertex belongs to exactly two of the sets $W_a \cup W_{a+1}$. Hence, for some $a \in \{0, \ldots, d-1\}$ we have $|W_a \cup W_{a+1}| \leqslant 2n/d \leqslant 2n^{2/3}$. ⌟

Partition the edges of G into E_1 and E_2 as follows:
- E_1 comprises all edges with one endpoint in W_a and the other in W_{a+1};
- E_2 comprises all the remaining edges.

Next, define subgraphs G_1 and G_2 of G as follows:

$$G_1 = (W_a \cup W_{a+1}, E_1) \qquad \text{and} \qquad G_2 = (V, E_2).$$

We first show that G_1 is a very simple and small graph.

Claim 6. *The graph G_1 has at most $2n^{2/3}$ vertices and treewidth at most $2w + 1$.*

PROOF. The bound on the number of vertices of G_1 is directly implied by Claim 5.

For the treewidth bound, note that every connected component of G_1 is a subgraph of the graph $G[L_i \cup L_{i+1}]$ for some $i \in \mathbb{N}$. Next, observe that mapping φ restricted to $L_i \cup L_{i+1}$ witnesses that $G[L_i \cup L_{i+1}]$ is a subgraph of $H \boxtimes K_2$, where K_2 is the graph consisting of two adjacent vertices. It is easy to see that if H has treewidth at most w, then $H \boxtimes K_2$ has treewidth at most $2w + 1$: in a tree decomposition of H of width at most w just replace every vertex v with the two copies of v in $H \boxtimes K_2$. Since treewidth does not grow under taking subgraphs, we conclude that $G[L_i \cup L_{i+1}]$ has treewidth at most $2w + 1$. Hence every connected component of G_1 has treewidth at most $2w + 1$, so we can conclude the same about the whole G_1. ⌟

We now analyze the graph G_2. The idea is to apply Lemma 6, so we need to find a subgraph embedding of G_2 into a graph $H' \boxtimes P'$, where H' has bounded treewidth while P' is a *short* path. We do it as follows.

Claim 7. *In polynomial time one can compute a graph H' of treewidth at most w and a subgraph embedding φ' of G_2 into $H' \boxtimes P'$, where P' is a path of length $d - 1$. Moreover, we have*

$$W_a = \varphi'^{-1}(\{(v, d-1) \colon v \in V(H')\}) \qquad \text{and} \qquad W_{a+1} = \varphi'^{-1}(\{(v, 0) \colon v \in V(H')\}).$$

PROOF. Let H' be the graph obtained by taking n disjoint copies of H; clearly the treewidth of H' is at most w. We assume that every vertex of H' is represented as a pair (v, j), where $v \in V(H)$ and $j \in \{0, \ldots, n-1\}$ is the index of the copy of H in H'. Let P' be the path of length $d-1$, whose vertices are indexed with numbers $0, 1, \ldots, d-1$. Consider the following mapping φ' from $V(G)$ to $V(H' \boxtimes P')$: for $u \in V(G)$, denoting $(v, i) = \varphi(u)$ and $j = i + d - a - 1$, we set

$$\varphi'(u) = (\, (v\,,\, j \text{ div } d)\,,\, j \bmod d\,),$$

where j div $d = \lfloor j/d \rfloor$. Note that the assertion that $W_a = \varphi'^{-1}(\{(v, d-1) : v \in V(H')\})$ and $W_{a+1} = \varphi'^{-1}(\{(v, 0) : v \in V(H')\})$ follows directly from the definition.

It is straightforward to verify that φ' is a subgraph embedding from G_2 to $H' \boxtimes P'$: the subgraph $G_2[L_0 \cup \ldots \cup L_a]$ is mapped to the first copy of H (times P'), the subgraph $G_2[L_{a+1} \cup \ldots \cup L_{a+d}]$ is mapped to the second copy of H (times P'), and so on. Note here that in G_2 there are no edges between L_a and L_{a+1}, nor between L_{d+a} and L_{d+a+1}, and so on. ⌋

We can now use Claims 6 and 7 to give the promised labeling scheme. First, by Claim 6 and Theorem 2, for the graph G_1 we may compute a labeling λ_1 with labels of length at most $\frac{2}{3}\log n + O(\log \log n)$. Second, by Claim 7 and Lemma 6, for G_2 we may compute[4] a labeling λ_2 with labels of length at most $\log n + \log d + O(\log \log n) = \frac{4}{3}\log n + O(\log \log n)$. Moreover, we may construct this labeling so that all vertices of $W_a \cup W_{a+1}$ receive shorter labels, namely of length at most $\frac{2}{3}\log n + O(\log \log n)$. Here, we use Remark 1 together with the second statement of Claim 7 in order to reduce the $\log d$ summand to $O(1)$, and we use $Q = W_a \cup W_{a+1}$ as the prescribed set of at most $2n^{2/3}$ vertices in order to reduce the $\log n$ summand to $\frac{2}{3}\log n + O(1)$.

Now, for any vertex u of G, we define its label $\lambda(u)$ as follows:

- If $u \notin W_a \cup W_{a+1}$, then $\lambda(u) = \lambda_2(u)$.

- If $u \in W_a \cup W_{a+1}$, then $\lambda(u)$ is the concatenation of $\lambda_1(u)$ and $\lambda_2(u)$.

In the second case, in order to be able to decode $\lambda_1(u)$ and $\lambda_2(u)$ from $\lambda(u)$, we append $\log \log n$ bits that indicate the length of $\lambda_1(u)$. Also, we add one bit indicating the case into which the vertex u falls.

Thus, in the first case $\lambda(u)$ is of length $\frac{4}{3}\log n + O(\log \log n)$, while in the second it is of length

$$\frac{2}{3}\log n + O(\log \log n) + \frac{2}{3}\log n + O(\log \log n) + \log \log n = \frac{4}{3}\log n + O(\log \log n)\,.$$

Hence, the length of the labeling scheme is as claimed. For the implementation of the Decoder, given labels of two vertices u and u', we simply read labels of u and u' in λ_2 and λ_1 (if applicable) and check whether they are adjacent either in G_1 or in G_2. This concludes the construction of the labeling scheme.

The above construction can be directly translated to an implementation of the Encoder in polynomial time and the Decoder in constant time. In case of Claim 5, note that an index a satisfying the claim can be found in polynomial time by checking all the integers between 0 and $d-1$ one by one. □

Remark 3. Note that the statement of Theorem 1 considers the class $\mathcal{C}$ fixed, hence the factors hidden in the $O(\cdot)$ notation depend on the constant w given by the efficient flatness of $\mathcal{C}$. It is not hard to verify that the obtained dependence on w is linear, that is, the length of the labeling scheme provided by Theorem 1 is $\frac{4}{3}\log n + c \cdot w \cdot \log \log n$ for some absolute constant $c \in \mathbb{N}$.

---

[4]A careful reader might be worried at this point that the graph H' produced by Claim 7 may have as many as $\Omega(n^2)$ vertices. However, since G has at most n vertices, we may again remove vertices of H' that do not participate in the image of G under φ', thus bringing $|V(H')|$ to at most n. In fact, this is what happens at the beginning of the proof of Lemma 6.

6 Conclusions

We gave an upper bound of $(\frac{4}{3} + o(1)) \log n$ for the length of labeling schemes for any efficiently flat class of graphs. This result applies to the class of planar graphs — which were our main motivation — but also encompasses more general classes such as graphs embeddable in a fixed surface, or k-planar graphs for any fixed k. Subsequent to our work, Dujmović et al. [DEJ$^+$20] announced a construction of labeling schemes for any efficiently flat class of graphs with the optimum length $(1 + o(1)) \log n$. Their approach also relies on the structure theorem of [DJM$^+$19], but is much more involved than the one presented here. Also, it leads to a non-constant-time implementation of the Decoder.

So far, the extent of these results is delimited by the flatness of the considered classes of graphs. As discussed in [DJM$^+$19], this property is unfortunately not enjoyed by every proper minor-closed graph class, but holds for every class of nearly embeddable graphs without apices (formally, $(0, g, k, p)$-nearly embeddable graphs for fixed $g, k, p \in \mathbb{N}$). Such graphs are the basic building blocks in the Structure Theorem for proper minor-closed classes of Robertson and Seymour [RS04]. This gives hope for extending the existence of labeling schemes of length $(1 + o(1)) \log n$ to all proper minor-closed classes through the Structure Theorem. In fact, such a line of reasoning was successfully applied in [DJM$^+$19] to obtain upper bounds on the queue number in proper minor-closed classes. In the case of labeling schemes, combining the labelings along a tree decomposition into nearly-embeddable parts seems to be the main issue.

References

[AABTKS16] M. Abrahansen, S. Alstrup, M. Bæk Tejs Knudsen, and M. Stöckel, *Near-optimal induced universal graphs for cycles and paths*, Tech. Rep. 1607.04911v2 [cs.DS], arXiv, July 2016.

[ACGP16] I. Abraham, S. Chechik, C. Gavoille, and D. Peleg, *Forbidden-set distance labels for graphs of bounded doubling dimension*, ACM Transactions on Algorithms, 12 (2016), pp. 22:1–22:17. DOI: 10.1145/2818694.

[ACJR19] H. Acan, S. Chakraborty, S. Jo, and S. S. Rao, *Succinct data structures for families of interval graphs*, Tech. Rep. 1902.09228v1 [cs.DS], arXiv, February 2019.

[ADBTK17] S. Alstrup, S. Dahlgaard, and M. Bæk Tejs Knudsen, *Optimal induced universal graphs and adjacency labeling for trees*, Journal of the ACM, 64 (2017), pp. 27:1–27:22. DOI: 10.1145/3088513.

[AGHP16] S. Alstrup, C. Gavoille, E. B. Halvorsen, and H. Petersen, *Simpler, faster and shorter labels for distances in graphs*, in 27$^{\text{th}}$ Symposium on Discrete Algorithms, SODA 2016, ACM-SIAM, January 2016, pp. 338–350. DOI: 10.1137/1.9781611974331.ch25.

[AGKR04]   S. Alstrup, C. Gavoille, H. Kaplan, and T. Rauhe, *Nearest common ancestors: A survey and a new algorithm for a distributed environment*, Theory of Computing Systems, 37 (2004), pp. 441–456. doi: 10.1007/s00224-004-1155-5.

[AHGL14]   S. Alstrup, E. B. Halvorsen, and K. Green Larsen, *Near-optimal labeling schemes for nearest common ancestors*, in 25th Symposium on Discrete Algorithms, SODA 2014, ACM-SIAM, January 2014, pp. 972–982. doi: 10.1137/1.9781611973402.72.

[AKTZ15]   S. Alstrup, H. Kaplan, M. Thorup, and U. Zwick, *Adjacency labeling schemes and induced-universal graphs*, in 47th Annual ACM Symposium on Theory of Computing, STOC 2015, ACM Press, June 2015, pp. 625–634. doi: 10.1145/2746539.2746545.

[AKTZ19]   S. Alstrup, H. Kaplan, M. Thorup, and U. Zwick, *Adjacency labeling schemes and induced-universal graphs*, SIAM Journal on Discrete Mathematics, 33 (2019), pp. 116–137.   doi: 10.1137/16M1105967.

[Alo17]   N. Alon, *Asymptotically optimal induced universal graphs*, Geometric and Functional Analysis, 27 (2017), pp. 1–32. doi: 10.1007/s00039-017-0396-9.

[AR02]   S. Alstrup and T. Rauhe, *Small induced-universal graphs and compact implicit graph representations*, in 43rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2002, IEEE Computer Society Press, November 2002, pp. 53–62.   doi: 10.1109/SFCS.2002.1181882.

[AR14]   D. Adjiashvili and N. Rotbart, *Labeling schemes for bounded degree graphs*, in 41st International Colloquium on Automata, Languages and Programming, ICALP 2014, vol. 8573 of Lecture Notes in Computer Science (ARCoSS), Springer, July 2014, pp. 375–386.   doi: 10.1007/978-3-662-43951-7_32.

[BCE+82]   L. Babai, F. R. K. Chung, P. Erdős, R. L. Graham, and J. H. Spencer, *On graphs which contain all sparse graphs*, Annals of Discrete Mathematics, 12 (1982), pp. 21–26.   doi: 10.1016/S0304-0208(08)73486-8.

[BGL06]   N. Bonichon, C. Gavoille, and A. Labourel, *Short labels by traversal and jumping*, in 13th International Colloquium on Structural Information & Communication Complexity, SIROCCO 2006, vol. 4056 of Lecture Notes in Computer Science, Springer, July 2006, pp. 143–156. doi: 10.1007/11780823_12.

[BGP20]   M. Bonamy, C. Gavoille, and M. Pilipczuk, *Shorter labeling schemes for planar graphs*, in 31st Symposium on Discrete Algorithms, SODA 2020, SIAM, 2020, pp. 446–462.   doi: 10.1137/1.9781611975994.27, https://doi.org/10.1137/1.9781611975994.27.

[Bod96]   H. L. Bodlaender, *A linear time algorithm for finding tree-decompositions of small treewidth*, SIAM Journal on Computing, 25 (1996), pp. 1305–1317. doi: 10.1137/S0097539793251219.

[But09]   S. Butler, *Induced-universal graphs for graphs with bounded maximum degree*, Graphs and Combinatorics, 25 (2009), pp. 461–468. doi: 10.1007/s00373-009-0860-x.

[Cap02]   M. R. Capalbo, *Small universal graphs for bounded-degree planar graphs*, Combinatorica, 22 (2002), pp. 345–359. doi: 10.1007/s004930200017.

[CGO10] J. Chalopin, D. Gonçalves, and P. Ochem, *Planar graphs have 1-string representations*, Discrete & Computational Geometry, 43 (2010), pp. 626–647. DOI: 10.1007/s00454-009-9196-9.

[Chu90] F. R. K. Chung, *Universal graphs and induced-universal graphs*, Journal of Graph Theory, 14 (1990), pp. 443–454. DOI: 10.1002/jgt.3190140408.

[DEJ+19] V. Dujmović, L. Esperet, G. Joret, B. Walczak, and D. R. Wood, *Planar graphs have bounded nonrepetitive chromatic number*, Tech. Rep. 1904.05269v1 [math.CO], arXiv, 2019.

[DEJ+20] V. Dujmović, L. Esperet, G. Joret, C. Gavoille, P. Micek, and P. Morin, *Adjacency labelling for planar graphs (and beyond)*, Tech. Rep. 2003.04280 [math.CO], arXiv, March 2020.

[dFOdMR94] H. de Fraysseix, P. Ossona de Mendez, and P. Rosenstiehl, *On triangle contact graphs*, Combinatorics, Probability & Computing, 3 (1994), pp. 233–246. DOI: 10.1017/S0963548300001139.

[DHJ+20] Z. Dvořák, T. Huynh, G. Joret, C. Liu, and D. R. Wood, *Notes on graph product structure theory*, Tech. Rep. 2001.08860 [math.CO], arXiv, January 2020.

[DJM+19] V. Dujmović, G. Joret, P. Micek, P. Morin, T. Ueckerdt, and D. R. Wood, *Planar graphs have bounded queue-number*, in 60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, IEEE Computer Society, 2019, pp. 862–875. DOI: 10.1109/FOCS.2019.00056. see also arXiv:1904.04791v2 [cs.DM].

[DMW19] V. Dujmović, P. Morin, and D. R. Wood, *The structure of k-planar graphs*, Tech. Rep. 1907.05168 [math.CO], arXiv, July 2019.

[ELO08] L. Esperet, A. Labourel, and P. Ochem, *On induced-universal graphs for the class of bounded-degree graphs*, Information Processing Letters, 108 (2008), pp. 255–260. DOI: 10.1016/j.ipl.2008.04.020.

[FG01] P. Fraigniaud and C. Gavoille, *Routing in trees*, in 28th International Colloquium on Automata, Languages and Programming, ICALP 2001, vol. 2076 of Lecture Notes in Computer Science, Springer, July 2001, pp. 757–772. DOI: 10.1007/3-540-48224-5_62.

[FGNW17] O. Freedman, P. Gawrychowski, P. K. Nicholson, and O. Weimann, *Optimal distance labeling schemes for trees*, in 36th Annual ACM Symposium on Principles of Distributed Computing, PODC 2017, ACM Press, July 2017, pp. 185–194. DOI: 10.1145/3087801.3087804.

[FK10a] P. Fraigniaud and A. Korman, *Compact ancestry labeling schemes for XML trees*, in 21st Symposium on Discrete Algorithms, SODA 2010, ACM-SIAM, January 2010, pp. 458–466. DOI: 10.1137/1.9781611973075.38.

[FK10b] P. Fraigniaud and A. Korman, *An optimal ancestry scheme and small universal posets*, in 42nd Annual ACM Symposium on Theory of Computing, STOC 2010, ACM Press, June 2010, pp. 611–620. DOI: 10.1145/1806689.1806773.

[GIP18] D. Gonçalves, L. Isenmann, and C. Pennarun, *Planar graphs as L-intersection or L-contact graphs*, in 29th Symposium on Discrete Algorithms, SODA 2018, ACM-SIAM, 2018, pp. 172–184. DOI: 10.1137/1.9781611975031.12.

[GKU16]   P. Gawrychowski, A. Kosowski, and P. Uznański, *Sublinear-space distance labeling using hubs*, in 30th International Symposium on Distributed Computing, DISC 2016, vol. 9888 of Lecture Notes in Computer Science, Springer, September 2016, pp. 230–242. DOI: 10.1007/978-3-662-53426-7_17.

[GL07]   C. Gavoille and A. Labourel, *Shorter implicit representation for planar graphs and bounded treewidth graphs*, in 15th Annual European Symposium on Algorithms, ESA 2007, vol. 4698 of Lecture Notes in Computer Science, Springer, October 2007, pp. 582–593. DOI: 10.1007/978-3-540-75520-3_52.

[GP03]   C. Gavoille and D. Peleg, *Compact and localized distributed data structures*, Distributed Computing, 16 (2003), pp. 111–120. DOI: 10.1007/s00446-002-0073-5. PODC 20-Year Special Issue.

[GP08]   C. Gavoille and C. Paul, *Optimal distance labeling for interval graphs and related graphs families*, SIAM Journal on Discrete Mathematics, 22 (2008), pp. 1239–1258. DOI: 10.1137/050635006.

[GU16]   P. Gawrychowski and P. Uznański, *A note on distance labeling in planar graphs*, Tech. Rep. 1611.06529v1 [cs.DS], arXiv, November 2016.

[KKKP04]   M. Katz, N. A. Katz, A. Korman, and D. Peleg, *Labeling schemes for flow and connectivity*, SIAM Journal on Computing, 34 (2004), pp. 23–40. DOI: 10.1137/S0097539703433912.

[KNR88]   S. Kannan, M. Naor, and S. Rudich, *Implicit representation of graphs*, in 20th Annual ACM Symposium on Theory of Computing, STOC 1988, ACM Press, May 1988, pp. 334–343. DOI: 10.1145/62212.62244.

[Koe36]   P. Koebe, *Kontaktprobleme der konformen abbildung*, Berichte über die Verhandlungen der Sächsische Akademie der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse, 88 (1936), pp. 141–164.

[Pel00]   D. Peleg, *Informative labeling schemes for graphs*, in 25th International Symposium on Mathematical Foundations of Computer Science, MFCS 2000, vol. 1893 of Lecture Notes in Computer Science, Springer, August 2000, pp. 579–588. DOI: 10.1007/3-540-44612-5_53.

[Pel05]   D. Peleg, *Informative labeling schemes for graphs*, Theoretical Computer Science, 340 (2005), pp. 577–593. DOI: 10.1016/j.tcs.2005.03.015.

[RLDL94]   C. J. Rhee, Y. D. Liang, S. K. Dhall, and S. Lakshmivarahan, *Efficient algorithms for finding depth-first and breadth-first search trees in permutation graphs*, Information Processing Letters, 49 (1994), pp. 45–50. DOI: 10.1016/0020-0190(94)90053-1.

[Rot16]   N. Rotbart, *New Ideas on Labeling Schemes*, PhD thesis, University of Copenhagen, October 2016. http://www.academia.edu/33855491/New_Ideas_on_Labeling_Schemes.

[RS04]   N. Robertson and P. D. Seymour, *Graph minors. XX. Wagner's conjecture*, Journal of Combinatorial Theory, Series B, 92 (2004), pp. 325–357. DOI: 10.1016/j.jctb.2004.08.001.

[RT15]   L. Roditty and R. Tov, *New routing techniques and their applications*, in 34th Annual ACM Symposium on Principles of Distributed Computing, PODC 2015, ACM Press, July 2015, pp. 23–32. DOI: [10.1145/2767386.2767409](https://doi.org/10.1145/2767386.2767409).

[Sch89]   W. Schnyder, *Planar graphs and poset dimension*, Order, 5 (1989), pp. 323–343. DOI: [10.1007/BF00353652](https://doi.org/10.1007/BF00353652).

[Spi03]   J. P. Spinrad, *Efficient Graph Representations*, vol. 19 of Fields Institute Monographs, American Mathematical Society, 2003.

[Tho86]   C. Thomassen, *Interval representations of planar graphs*, Journal of Combinatorial Theory, Series B, 40 (1986), pp. 9–20. DOI: [10.1016/0095-8956(86)90061-4](https://doi.org/10.1016/0095-8956(86)90061-4).

[TZ01]   M. Thorup and U. Zwick, *Compact routing schemes*, in 13th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001, ACM Press, July 2001, pp. 1–10. DOI: [10.1145/378580.378581](https://doi.org/10.1145/378580.378581).