Shorter Implicit Representation for Planar Graphs and Bounded Treewidth Graphs

Cyril Gavoille* and Arnaud Labourel

LaBRI, Université de Bordeaux, France
{gavoille,labourel}@labri.fr

Abstract. Implicit representation of graphs is a coding of the structure of graphs using distinct labels so that adjacency between any two vertices can be decided by inspecting their labels alone. All previous implicit representations of planar graphs were based on the classical three forests decomposition technique (a.k.a. Schnyder's trees), yielding asymptotically to a $3\log n$-bit[1] label representation where n is the number of vertices of the graph.

We propose a new implicit representation of planar graphs using asymptotically $2\log n$-bit labels. As a byproduct we have an explicit construction of a graph with $n^{2+o(1)}$ vertices containing all n-vertex planar graphs as induced subgraph, the best previous size of such induced-universal graph was $O(n^3)$.

More generally, for graphs excluding a fixed minor, we construct a $2\log n + O(\log\log n)$ implicit representation. For treewidth-k graphs we give a $\log n + O(k\log\log(n/k))$ implicit representation, improving the $O(k\log n)$ representation of Kannan, Naor and Rudich [18] (STOC '88).

Our representations for planar and treewidth-k graphs are easy to implement, all the labels can be constructed in $O(n\log n)$ time, and support constant time adjacency testing.

1 Introduction

How to represent a graph in memory is a basic and fundamental data structure question. The two basic ways are adjacency matrices and adjacency lists. The latter representation is space efficient for sparse graphs, but adjacency queries require searching in the list, whereas matrices allow fast queries to the price of a super-linear space. Another technique, called *implicit representation* or *adjacency labeling scheme*, consists in assigning distinct labels to each vertex such that adjacency queries can be computed alone from the labels of the two involved vertices without any extra information source. So the graph can be manipulated by keeping only its labels in memory, any other global information on the graph (like its matrix) can be removed. The goal is to minimize the maximum length of a label associated with a vertex while keeping fast adjacency queries.

* Both authors are supported by the ANR-projects GEOCOMP and GRAAL.

[1] All logarithms are in base two.

L. Arge and E. Welzl (Eds.): ESA 2007, LNCS 4698, pp. 582–593, 2007.

The notion of adjacency labeling scheme is closely related to that of induced-universal graphs, introduced by [12] for VLSI circuit design. A graph $\mathcal{U}$ is said to be *induced-universal* for a given graph family $\mathcal{F}$, if every graph of $\mathcal{F}$ is isomorphic to some induced subgraph of $\mathcal{U}$. Kannan, Naor and Rudich [18] established that there is an adjacency labeling scheme with labels of k bits for $\mathcal{F}$ if and only if there exists an induced-universal graph with 2^k vertices. Therefore, the combinatorial problem of constructing a small induced-universal graph for $\mathcal{F}$ is equivalent in designing a labeling scheme with short labels for every graph of $\mathcal{F}$, any result on one of the problems transferring on the other. For instance, the best known induced-universal graph for the class of n-node forests, namely $n \cdot 2^{O(\log^* n)}$, is actually derived from a labeling scheme with $\log n + O(\log^* n)$-bit[2] labels [3].

We note that a labeling scheme transfers to an effective construction of the corresponding induced-universal graph: the vertex-set of the universal graph is the set of all possible labels, and an edge is added if the adjacency test between the two corresponding vertices/labels is positive. However, converting a small induced-universal graph into a compact labeling scheme does not give, in general, any efficient representation in term of computation of all the labels for a graph and adjacency testing time.

1.1 Related Works

Motivated by applications in XML search engines, and distributed applications as peer-to-peer overlay networks or network routing, several queries on distributed data-structures have been investigated recently. In this framework, distributed data-structure can be seen as a label assigned to each node such that queries can be answered by inspecting the labels only, without any other source of information. For instance, address-based routing in trees [13,19,25], distance queries for interval and permutation graphs [5,14], sibling and ancestry in rooted trees [1], etc. have $O(\log n)$-bit distributed data-structures.

Representation of graphs with short labels, introduced by Breuer [8], have been investigated by Kannan, Naor and Rudich [18]. They construct adjacency labeling schemes for several families of graphs including treewidth[3]-k and arboricity[4]-α graphs using respectively $O(k \log n)$-bit and $(\alpha+1)\log n$-bit labels. In particular, this latter scheme gives an induced-universal graph for trees ($\alpha = 1$) of size $O(n^2)$, and combined with the linear time Schnyder's tree decomposition [24], this leads to the first effective $4 \log n$-bit labeling for planar graphs ($\alpha = 3$).

The size of the induced-universal graph for trees was later improved in a non-trivial way to $O(n \log n)$ by Chung [9]. This transfers to a non-constructive adjacency labeling with labels of $\log n + \log\log n + O(1)$ bits for trees, and more generally of $\alpha \log n + O(\alpha \log\log n)$ bits for arboricity-α graphs. This result has

[2] $\log^* n$ denotes the number of times log should be iterated to get a constant.

[3] The original result was stated for c-decomposable graphs which are exactly the graphs of treewidth $k = \Theta(c)$.

[4] Which is the smallest number of forests in which the edge-set of the graph can be partitioned.

been further improved by the use of an efficient labeling scheme to $\alpha \log n + O(\log^* n)$ bits [3], complemented with a $\alpha \log n - O(\alpha^2)$ lower bound. This best to date result yields to an adjacency labeling scheme for trees of $\log n + O(\log^* n)$. For planar graphs, the resulting $3 \log n + O(\log^* n)$-bit labeling can be slightly improved by observing that: 1) planar graphs can be decomposed into three forests those one is spanning and of bounded degree [17]; and 2) adjacency in bounded degree forests can be done efficiently in $\log n + O(1)$ [7]. Thus, a vertex can store its label in the bounded degree forest plus the label of its parent in each of the two other forests, leading to a $3 \log n + O(1)$ labeling with constant time adjacency. This converts into an induced-universal graph of size $O(n^3)$.

Finally, we remark that the best up to date planar representation essentially uses the three forests decomposition.

1.2 Our Contributions

1. Surpassing the three forests decomposition of planar graph, we propose a new $2 \log n + O(\log \log n)$ bit adjacency labeling scheme supporting constant query time.
2. More generally, for graphs excluding a fixed graph H as minor[5] (e.g., planar graphs exclude K_5), we propose a $2 \log n + O(h \log \log(n/h))$ bit labeling scheme where h is a constant only depending on H.
3. For treewidth-k graphs, a subclass of graphs excluding K_{k+2}-minor, we improve the label length to $\log n + O(k \log \log(n/k))$ supporting adjacency testing in constant time (independent of k).
4. We also show that every adjacency labeling scheme for treewidth-k graphs requires labels of $\log n + \Omega(k)$ bits, proving that the linearity dependency on k in the second order term of our scheme is necessary.
5. All these results transfer also to effective construction of smaller induced-universal graphs of size $n^{2+o(1)}$ for minor-free graphs, and of $n^{1+o(1)}$ for bounded treewidth graphs, previous bounds were $O(n^3)$ for planar graphs and $n^{O(k)}$ for treewidth-k graphs.

1.3 Techniques

In fact our results on planar and minor-free graphs are derived from our $(1 + o(1)) \log n$-bit label scheme for bounded treewidth graphs using suitable edge-partitions. Although planar graphs can be decomposed into three treewidth-1 graphs (forests), they can also be decomposed, in linear time as well, in two bounded treewidth graphs as follows: 1) partition the vertices into layers L_i at distance i from an arbitrary vertex; 2) the graph H_i, $i > 0$, induced by the edge $\{u, v\}$ with $u, v \in L_i$ or with $u \in L_i$ and $v \in L_{i-1}$ has bounded treewidth (actually at most three from [11]); 3) the two graphs composed as the union of the H_i's for respectively odd and even i are therefore of bounded treewidth since no edges lie between H_i and H_{i+2}.

[5] That are graphs for which H can be obtained by edge contractions and taking subgraphs.

From the above discussion, a λ-bit labeling for treewidth-k graphs yields a 2λ-bit labeling scheme for planar graphs, because it suffices to assign to each vertex the label for each subgraph and to test adjacency in the two subgraphs. Actually, it has been recently proved that planar graphs can be edge-partitioned into two outerplanars [16] (that are of treewidth two), and in linear time [17]. It has been proved a similar result for graphs excluding a fixed minor [10]: they can be edge-partitioned into two subgraphs of treewidth only depending on the excluded minor.

Therefore, our $\log n + O(\log \log n)$ labeling scheme for bounded treewidth graphs implies a $2 \log n + O(\log \log n)$ not only for planar graphs but also for all minor-free graphs (including bounded genus graphs for instance). All previous schemes were based on the bounded arboricity of bounded treewidth or minor-free graphs. Unfortunately, arboricity-α graphs require labels of at least $\alpha \log n - O(\alpha^2)$ bits from the lower bound of [3], and outerplanar graphs (which exclude K_4) have already arboricity two. So there is no advantage of decomposing a graph into low arboricity subgraphs, unlike decomposing into low treewidth subgraphs: 2 treewidth-2 graphs is better than 3 treewidth-1 graphs.

From the above discussion, we detail our scheme only for treewidth-k graphs. A graph has *treewidth* k if and only if it is a subgraph of a chordal (or triangulated) graph[6] of maximum clique size $k + 1$. Chordal graphs, and therefore treewidth-k graphs, have a natural clique tree structure, and they can also be defined as graphs having a Robertson-Seymour's tree-decomposition [23] in subgraphs of at most $k + 1$ vertices, called *bags*. Informally, the set of bags forms a cover of the graph (each vertex and edge belongs to at least one bag), and they must be one-to-one mapped to the nodes of a tree T such that the set of bags containing a given vertex of the graph induces a connected component of T.

Let us first observe that an efficient labeling scheme on trees (like the one of [3]) does not transfer to a labeling scheme for graphs having some "good" tree-decomposition because bags may intersect in a non trivial way.

The next section present our labeling scheme. Due to lack of space the lower bound appears in the full version.

2 A Simple Adjacency Labeling Scheme

In this section we prove the main result of this paper that is:

Theorem 1. *The family of n-vertex treewidth-k graphs enjoys an adjacency labeling scheme with labels of $\log n + O(k \log \log(n/k))$ bits, and with a constant adjacency query time. Moreover, for every fixed k, all the labels of such a graph can be computed in $O(n \log n)$ time.*

The construction of the labels relies on finding a chordal supergraph with minimum maximal clique (i.e., $k + 1$), whose the associated decision problem is NP-complete [4]. For each fixed k, linear time algorithms are known [6]. We

[6] I.e., a graph whose the length of the longest induced cycle is at most three.

also note that for trees, outerplanar and series-parallel graphs (i.e., for $k \leqslant 2$) efficient and simple algorithms exist, and our compact representation for planar graph is based on outerplanar graphs only.

As said previously, the results for the $2\log n + O(\log\log n)$ bit labeling for planar graphs, and more generally minor-free graphs, and the small induced-universal graphs, are simple corollaries of Theorem 1.

2.1 Preliminaries

Our scheme relies on the chordal representation: every treewidth-k graph is the spanning subgraph of a chordal graph, called *triangulation*, whose maximal cliques have no more than $k+1$ vertices. Chordal graphs of maximum clique size $k+1$ have two important properties. Firstly, there exists a clique whose its removal leaves the graph with connected components at most half the size of the original graph [15]. Secondly, there is an elimination ordering scheme of the vertices, called *simplicial elimination scheme*, such that the neighborhood of each just removed vertex is a clique in the remaining graph. Our scheme strongly relies on graphs having these both properties we formalize hereafter.

A graph G is *s-separable* if every subgraph H has a half-separator of size at most s, i.e., a subset of at most s vertices whose removal leaves H in connected components of at most $|V(H)|/2$ vertices. A graph is *k-clique orientable* if its edges can be oriented such that the out-neighborhood of every vertex induces a clique of size at most k. Such an orientation is called a k-clique orientation.

Definition 1. *A* (k,s)-triangulation *is a graph that is k-clique orientable and s-separable. A* (k,s)-graph *is a subgraph of a* (k,s)*-triangulation.*

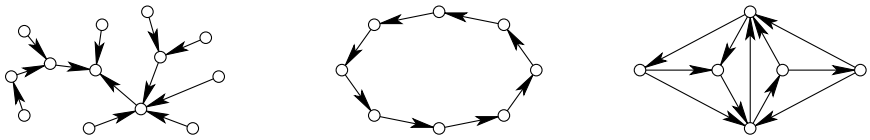

Fig. 1. A $(1,2)$-triangulation, a $(1,2)$-triangulation, and a $(1,2)$-triangulation

According to this definition, treewidth-k are $(k,k+1)$-graphs, since their triangulations are $(k+1)$-separable and the simplicial elimination scheme provides a k-clique orientation. However, forests are $(1,1)$-graphs whereas they are of treewidth one. Cycles (treewidth 2) are 1-clique orientable and 2-separable, and thus are $(1,2)$-graphs (they are also $(1,2)$-triangulations, see Fig. 1). We can also check that cliques of k vertices are $(\lceil k/2\rceil, \lceil k/2\rceil)$-graphs whereas they are of treewidth $k-1$. So, $(k,k+1)$-graphs are a strictly wider family of graphs than treewidth-k graphs.

A *k-complex* of a graph G is a subgraph K isomorphic to depth-1 tree with at most k leaves. Its *root*, denoted by $\mathrm{root}(K)$, is the non-leaf node of K. Two complexes K and K' of G are said *adjacent* if $\mathrm{root}(K) \neq \mathrm{root}(K')$, and if either

root$(K) \in V(K')$ or root$(K') \in V(K)$. Clearly, if two complexes are adjacent, then their roots are adjacent in G.

With each vertex u of a (k, s)-graph G one can associate a k-complex K_u of root u such that each edge of G is covered by exactly one k-complex, as follows: K_u is formed by u and its neighbors in G that are out-neighbors in the (k, s)-triangulation of G. Then, there is a trivial adjacency labeling scheme consisting in labeling u by a coding $V(K_u)$. It is clear that u and v are adjacent if and only if their associated complexes K_u and K_v are. This trivially yields to labels of $(k+1)\lceil \log n \rceil$ bits. We shall see that a much better implementation (or coding) of complexes can be achieved using the fact that, in the triangulation of G, K_u is a clique.

For this purpose we need a particular partition of the vertices of G.

Definition 2. *A* bidecomposition *of a graph G is a rooted binary tree T where nodes, called* parts*, form a partition of $V(G)$ such that the parts of the endpoints of every edge of G are related*[7] *in T.*

The part of T containing vertex u is denoted by T_u. A straightforward observation from the definition of T is that the parts of all the vertices of a subgraph K are pairwise related if K is a clique.

2.2 Finding a Suitable Bidecomposition

Lemma 1. *Let $\tilde{G}$ be an s-separable graph of n vertices. Then, $\tilde{G}$ has a bidecomposition such that the parts of depth h have at most $s\lfloor \log(2n/s) - h \rfloor$ vertices.*

Moreover, such a bidecomposition can be computed in $O(sn \log(n/s))$ time if we assume that a half-separator of size s for any subgraph of $\tilde{G}$ can be computed in time linear in the size of the subgraph.

Proof. Let $\tilde{G}$ be an s-separable graph with n vertices. Since every subgraph of an s-separable graph is also s-separable, we essentially need to construct the root of the willing bidecomposition T, i.e., to prove the result for depth $h = 0$, and repeat recursively the process on the remaining subgraphs. In the following, the root of the bidecomposition is called *biseparator*.

The root B of T (the biseparator) and the partition (V_1, V_2) of $V(\tilde{G}) \setminus B$ are computed thanks to the following simple procedure called Biseparator (Algorithm 1), where Half-Separator(H, s) is a subroutine computing a half-separator of size at most s for the graph H.

At each iteration of the while-main of Biseparator the size of H is divided by at least two since all the resulting components of $H \setminus S$ are removed from H, and the remaining largest component is of size at most $|V(H)|/2$. So, there is at most $\log(n/s)$ iterations.

At each step the size of B increases by at most s vertices, and the final step (last statement) adds at most s vertices to B. Therefore, $|B| \leqslant s \cdot \lfloor \log(n/s) \rfloor + s = s \cdot \lfloor \log(2n/s) - h \rfloor$ with $h = 0$.

[7] Two nodes of a rooted tree are *related* if one is ancestor of the other.

Algorithm 1. BISEPARATOR

Input : an s-separable graph $\tilde{G}$

Output: A biseparator B for $\tilde{G}$ and the associated vertex partition (V_1, V_2)

$H := \tilde{G}$; $V_1 := V_2 := B := \varnothing$
while $|V(H)| > s$ **do**
    $S :=$ HALF-SEPARATOR(H, s); $H := H \setminus S$; $B := B \cup S$
    forall *connected component* C *of* H *except the greatest* **do**
        $H := H \setminus C$;
        if $|V_1| > |V_2|$ **then** $V_2 := V_2 \cup V(C)$ **else** $V_1 := V_1 \cup V(C)$
$B := B \cup H$

In order to insure the correctness of the bidecomposition, we show the following loop invariant.

$$(P): \quad |V_1|, |V_2| \leqslant n/2 \quad \text{and} \quad V(H) \cup V_1 \cup V_2 \cup B = V(\tilde{G})$$

It is straightforward to verify that (P) is true at the beginning of the main loop. Let us show that (P) remains true at the end of the imbricated loop. The loop invariant clearly remains true after computation of the half-separator and the statement $H := H \setminus S$ and $B := B \cup S$. Assume w.l.o.g. that $|V_1| \geqslant |V_2|$. We have $V(H) \cup V_1 \cup V_2 \subseteq V(\tilde{G})$ since (P) is true. We obtain the following relation for the size of the sets.

$$\begin{aligned}
|V(H)| + |V_1| + |V_2| &\leqslant n \\
|V(H)| &\leqslant n - |V_1| - |V_2| \leqslant n - 2|V_2| \quad (|V_1| \geqslant |V_2|) \\
|V(H)|/2 &\leqslant n/2 - |V_2| \\
|V(C)| &\leqslant n/2 - |V_2| \text{ (there is yet another larger component in } H) \\
|V(C)| + |V_2| &\leqslant n/2
\end{aligned}$$

The property (P) remains true at the end of the imbricated loop. The property (P) is loop invariant and so is true at the end of the main loop. We have that $|V_1|, |V_2| \leqslant n/2$. Moreover, there is no edge linking vertices of V_1 to vertices of V_2 since what we add to V_1 or V_2 is a full connected component of H.

The bidecomposition T of $\tilde{G}$ is obtained by applying recursively the process on the graphs induced by V_1 and V_2. We then link to B (the root of T) the resulting bidecompositions if there are non-empty.

Since the size of V_1 and V_2 are $\leqslant n/2$, by induction, the size of their biseparators (so parts of depth $h = 1$ in T) would be at most $s \cdot \lfloor \log(2(n/2)/s) \rfloor = s \cdot \lfloor \log(2n/s) - 1 \rfloor$. More generally, for an arbitrary depth $h \geqslant 0$, the parts are of size $s \cdot \left\lfloor \log(2(n/2^h)/s) \right\rfloor = s \cdot \lfloor \log(2n/s) - h \rfloor$ as claimed.

Before computing the time complexity, let us observe that every subgraph H of $\tilde{G}$ has $O(s|V(H)|)$ edges. Indeed, it is known [22] that every s-separable graph has treewidth at most $4s$. Moreover, treewidth-k n-vertex graphs have no more than kn edges, due to the simplicial elimination scheme of their minimal triangulation. It follows that H has at most $4s|V(H)|$ edges.

Let us show now that it takes a linear time to compute the biseparator B, the root of T. At each iteration of the main loop, a separator of H is computed in $O(|V(H)|+|E(H)|)$ time by assumption. The updates of V_1, V_2, and H are also determined in $O(|V(H)|+|E(H)|)$. We have seen that $|E(H)| \leqslant 4s|V(H)|$. At the i-th iteration of the main loop, $|V(H)| \leqslant n/2^i$. So, the i-th iteration takes $O(|V(H)|+|E(H)|) = O(sn/2^i)$ time. The time for computing the biseparator is therefore $\sum_{i=0}^{\log(n/s)} O(sn/2^j) \leqslant c \cdot sn$ for some constant $c > 0$.

In total, the bidecomposition T is computed recursively in time $t(n) \leqslant c \cdot sn + 2t(n/2)$ if $n > s$, and $t(n) = O(1)$ in if $n \leqslant s$, which is $t(n) = O(sn \log(n/s))$. □

2.3 The Labels

Let us fix a (k,s)-graph G with n vertices. So G has a spanning (k,s)-triangulation $\tilde{G}$ that is s-separable. Let T be a bidecomposition for $\tilde{G}$ satisfying Lemma 1. For every vertex u of G, let K_u be a complex rooted at u obtained by adding to u every incident edge leading to a neighbors of u in G that is an out-neighbor in $\tilde{G}$. By construction, $V(K_u)$ induces a clique of at most $k+1$ vertices in $\tilde{G}$. Hence, the parts of each vertex of K_u are pairwise related in T.

We define the function $\beta(h) = \sum_{i=0}^{h} s \lfloor \log(2n/s) - i \rfloor$. Intuitively, $\beta(h)$ represents the maximum number of vertices of G contained in the parts of a branch[8] of T of length $h+1$. We have also $\beta(h) = s(h+1)(\lfloor \log(2n/s) \rfloor - h/2)$.

Let X be a part of T at depth h. Te root of T is at depth $h = 0$. By Lemma 1, $|X| \leqslant \beta(h) - \beta(h-1)$. We denote by $\mathrm{path}(X)$ the binary word defining the unique path from the root of T to X. The length of $\mathrm{path}(X)$ is $|\mathrm{path}(X)| = h$. We associated with each $u \in X$, its *rank*, a unique integer $\mathrm{rank}(u) \in [0, |X|)$, and its *position*, defined by $\mathrm{pos}(u) = \mathrm{rank}(u) + \beta(h-1)$. The *apex* of u is the vertex a_u of K_u with maximum position, i.e., such that $\mathrm{pos}(a_u) = \max_{v \in V(K_u)} \mathrm{pos}(v)$.

Observe that the positions are relative to a branch of T: every pair of vertices whose parts are on the same branch have distinct positions, and thus the parts of any two vertices having the same positions cannot be related.

Let u be a vertex of G, and let a_u be the apex of K_u in T. The label of vertex u is defined by the following quadruple:

$$\mathrm{label}(u) = (\mathrm{path}(T_{a_u}), \mathrm{rank}(a_u), P_u, r_u)$$

where $P_u = \{\mathrm{pos}(v) \mid v \in V(K_u), v \neq a_u\}$ and $r_u = |\{p \in P_u \mid p < \mathrm{pos}(u)\}|$. Roughly speaking, P_u is the set of positions of all the vertices of K_u but its apex, and r_u is the rank of the root of K_u in P_u. So $r_u = 0$ if $\mathrm{pos}(u)$ is the smallest position in P_u; $r_u = |P_u|$ if $u = a_u$ is the apex itself.

Let $\mathrm{pos}(K_u) = \{\mathrm{pos}(v) \mid v \in V(K_u)\}$. It is not difficult to see that the complex K_u is uniquely defined by the pair $(\mathrm{pos}(K_u), \mathrm{path}(T_{a_u}))$, i.e., the set of its positions and the path leading to its apex. Indeed, as said previously, vertices lying on the same branch of T have pairwise distinct positions, and vertices lying on different branches can be identified from the path of their apices (that must therefore differ). The set $\mathrm{pos}(K_u)$ is not a field of $\mathrm{label}(u)$, P_u misses $\mathrm{pos}(a_u)$. However,

[8] A path that leads to the root of T.

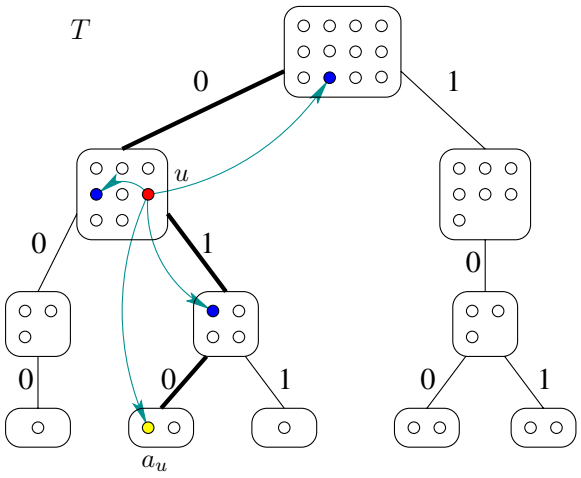

Fig. 2. A bidecomposition with a complex K_u of root u and apex a_u

it can be computed since $\mathrm{pos}(a_u) = \mathrm{rank}(a_u) + \beta(|\mathrm{path}(T_{a_u})|)$. It follows that $\mathrm{label}(u)$ is a (one-to-one) coding of K_u. In particular, $\mathrm{label}(u) \neq \mathrm{label}(v)$ since $K_u \neq K_v$ for distinct vertices $u \neq v$.

Lemma 2. *The labels are of* $\log n + 2k \log\log(n/s) + O(k \log(s/k))$ *bits.*

Proof. We assume that n, k, s are given. Let $w = \lfloor \log(2n/s) \rfloor$, and let $h = |\mathrm{path}(T_{a_u})|$. The binary word $\mathrm{path}(T_{a_u})$ is of length exactly h, and $\mathrm{rank}(a_u) \in [0, \beta(h) - \beta(h-1))$. We have $\beta(h) - \beta(h-1) = s \lfloor \log(2n/s) - h \rfloor = s(w-h)$.

We write $\mathrm{rank}(a_u) = x \cdot (w-h) + y$ where $x \in [0, s)$ and $y \in [0, w-h)$: $x = \lfloor \mathrm{rank}(a_u)/(w-h) \rfloor$ and $y = \mathrm{rank}(a_u) \bmod (w-h)$. We represent the two first fields $\mathrm{path}(T_{a_u}), \mathrm{rank}(a_u)$ by the binary string[9]:

$$S \;=\; 0^y \circ 1 \circ \sigma_x \circ \mathrm{path}(T_{a_u})$$

where σ_x denotes the standard binary representation of x on $\lceil \log s \rceil$ bits. Given S (but ignoring h), one can extract, y and σ_x, thus $\mathrm{path}(T_{a_u})$, h, and $\mathrm{rank}(a_u) = x \cdot (w-h) + y$. The length of S is $|S| \leqslant (w-h-1) + 1 + \lceil \log s \rceil + h = w + \lceil \log s \rceil = \lfloor \log(2n/s) \rfloor + \lceil \log s \rceil \leqslant \log n + O(1)$. All the positions in P_u range in $[0, \beta(h))$. It follows that there are at most $\binom{\beta(h)}{|P_u|} \leqslant \binom{\beta(h)}{k}$ possible ways to select the set P_u having at most k positions. This can be coded with $\log \binom{\beta(h)}{k} + O(1)$ bits. Finally, $r_u \in [0, |P_u|]$, thus there are $k+1$ possible values. This can be coded with $\lceil \log(k+1) \rceil$ bits. In total, the length of $\mathrm{label}(u)$ is at most:

$$|\mathrm{label}(u)| \;\leqslant\; \log n + \log \binom{\beta(h)}{k} + O(\log k) \qquad (1)$$

[9] We denote by $\circ$ the word concatenation.

By Lemma 1, the size of the parts in T of depth $\log(n/s)$ are of size at most $s \cdot (\log(2n/s) - \log(n/s)) = s$. From the while-condition in Algorithm BISEPARATOR, if $|V(H)| \leqslant s$, then the input graph is not separated at all, implying that the part is actually a leaf of T. Therefore $h \leqslant \log(n/s)$.

We have $\beta(h) \leqslant \beta(\log(n/s)) \leqslant (\log(n/s)+1) \cdot s \cdot \log(2n/s)) = s \cdot \log^2(2n/s)$. Thus $\log \binom{\beta(h)}{k} \leqslant k \log(\beta(h)/k) + O(k) \leqslant k \log(s/k \cdot \log^2(2n/s)) + O(k) \leqslant 2k \log\log(n/s) + O(k \log(s/k))$. Plugging this latter bound in Eq. (1), we get the desired result. □

So, for treewidth-k graphs, that are $(k, k+1)$-graphs, we obtain labels of length roughly $\log n + 2k \log\log(n/k)$. For $(1,1)$-graphs (forests), this is $\log n + 2\log\log n$. Actually, a finer analysis shows that for trees the label length is no more than $\log n + 2\log\log n + 2$ for every $n \geqslant 5$, a competitive bound with the $\log n + 4\log\log n$ scheme of [3].

We finally observe that our scheme support parent and sibling queries in rooted trees since we associate with each node a coding of itself and its parent. The label length is $\log n + O(\log\log n)$ which is not optimal for parent but optimal for sibling queries due to the lower bound of $\log n + \Omega(\log\log n)$ of [2].

2.4 Adjacency Test

Let u and v be two vertices of G. We denote by $M = \beta(h)$ where h is the length of the longest common prefix between $\text{path}(T_{a_u})$ and $\text{path}(T_{a_v})$. The proof of the next result appears in the full version.

Lemma 3. *The vertices u and v are adjacent if and only if* $\text{label}(u) \neq \text{label}(v)$ *and if either* $\text{pos}(u) \in \text{pos}(K_v) \cap [0, M)$ *or* $\text{pos}(v) \in \text{pos}(K_u) \cap [0, M)$.

We now briefly explain how to implement the adjacency test given by Lemma 3 to perform in constant time under the standard $\Omega(\log n)$-bit word RAM computer model. First we observe that $\beta(h)$ can be computed in constant time using its closed formula: $\beta(h) = s(h+1)(\lfloor s \log(2n/s) \rfloor - h/2)$. Thus M can be computed in constant time, because the length of the common prefix between two $O(\log n)$-bit words can be computed using constant number of MSB[10], binary masks, and shifting operations. Now, to test if $\text{pos}(v) \in \text{pos}(K_u)$, we can test if $\text{pos}(v) = \text{pos}(a_u)$ or if $\text{pos}(v) \in P_u$. The position of the apex is $\text{pos}(a_u) = \text{rank}(a_u) + \beta(|\text{path}(T_{a_u})|)$. The position of u can be also determined from $\text{label}(u)$, since $\text{pos}(u)$ is $\text{pos}(a_u)$ if $r_u = |P_u|$, or the r_u-th element of P_u. It follows that testing if $\text{label}(u) \neq \text{label}(v)$ can be done by checking that $(\text{pos}(u), \text{path}(T_{a_u})) \neq (\text{pos}(v), \text{path}(T_{a_v}))$. Finally, computing $\text{pos}(u)$ and testing whether $\text{pos}(u) \in P_v$ rely on membership and select query in an integer subset $P_u \subset \{0, \ldots, \beta(h)\}$.

Using a compact static dictionary, one can implement such queries in constant time using a data-structure of $(1+o(1)) \log \binom{\beta(h)}{|P_u|})$ bits [21]. We have already seen in the proof of Lemma 2 that $\log \binom{\beta(h)}{k} \leqslant 2k \log\log(n/s) + O(k \log(s/k))$.

[10] Most Significant Bit, or integer LOG function.

Lemma 4. *The family of n-vertex (k, s)-graphs enjoys an adjacency labeling scheme with label of $\log n + (2k + o(1)) \log\log(n/s)$ bits, and with a constant adjacency query time.*

We prove Theorem 1 by plugging $s = k+1$ and combining all lemmas, and using the fact that a $(k, k+1)$-triangulation for treewidth-k graphs can be computed in linear time for fixed k [6].

3 Concluding Remarks and Open Problems

We have proposed a new implicit representation for planar graphs, and more generally to graphs excluding a fixed minor, with asymptotically $2 \log n$ bits per vertex. Improving this bound would require a totally different approach, since decomposing into two or more subgraphs inevitably leads to a $c \log n$ representation with $c \geqslant 2$.

From the lower bound side, the number $\psi(n, H)$ of n-vertex labeled H-minor-free graphs is $n! \cdot 2^{hn+o(n)}$, where h depends only on H [20]. Therefore, the trivial information-theoretic lower bound for adjacency is $\frac{1}{n} \log \psi(n, H) \sim \log n + h$ bits for at least one label. This leads to the natural question we propose as open problem:

Conjecture. The family of H-minor-free graphs supports an adjacency labeling scheme with $\log n + h$ bit labels where $h = h(H)$.

Proving a labeling scheme of $\log n + O(k)$ for treewidth-k graphs would be already very interesting, since not only if would match our lower bound of $\log n + \Omega(k)$, but also prove an optimal bound for trees (up to an additive constant) which is still open.

References

1. Abiteboul, S., Alstrup, S., Kaplan, H., Milo, T., Rauhe, T.: Compact labeling schemes for ancestor queries. SIAM J. on Computing 35, 1295 (2006)
2. Alstrup, S., Bille, P., Rauhe, T.: Labeling schemes for small distances in trees. SIAM J. on Discrete Mathematics 19, 448–462 (2005)
3. Alstrup, S., Rauhe, T.: Small induced-universal graphs and compact implicit graph representations. In: 43^{rd} Annual IEEE Symp. on Foundations of Computer Science (FOCS), nov, pp. 53–62. IEEE Computer Society Press, Los Alamitos (2002)
4. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM J. on Algeb. and Disc. Meth. 8, 277–284 (1987)
5. Bazzaro, F., Gavoille, C.: Localized and compact data-structure for comparability graphs. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 1122–1131. Springer, Heidelberg (2005)
6. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. SIAM J. on Computing 25(6), 1305–1317 (1996)
7. Bonichon, N., Gavoille, C., Labourel, A.: Short labels by traversal and jumping. In: Flocchini, P., Gkasieniec, L. (eds.) SIROCCO 2006. LNCS, vol. 4056, pp. 143–156. Springer, Heidelberg (2006)

8. Breuer, M.A.: Coding the vertexes of a graph. IEEE Transactions on Information Theory IT-12, 148–153 (1966)
9. Chung, F.R.K.: Universal graphs and induced-universal graphs. J. of Graph Theory 14, 443–454 (1990)
10. DeVos, M., Ding, G., Oporowski, B., Sanders, D.P., Reed, B., Seymour, P.D., Vertigan, D.: Excluding any graph as a minor allows a low tree-width 2-coloring. J. of Combinatorial Theory, Series B 91(1), 25–41 (2004)
11. Elmallah, E.S., Colbourn, C.J.: Partitioning the edges of a planar graph into two partial k-trees. Congressus Numerantium 66, 69–80 (1988)
12. Erdös, P., Gerencsér, L., Máté, A.: Problems of graph theory concerning optimal design. In: Colloq. Math. Soc. Janos Bolyai 4: Combinatorial theory and its applications, vol. 1, pp. 317–325. North-Holland, Amsterdam (1970)
13. Fraigniaud, P., Gavoille, C.: Routing in trees. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 757–772. Springer, Heidelberg (2001)
14. Gavoille, C., Paul, C.: Optimal distance labeling schemes for interval and circular-arc graphs. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 254–265. Springer, Heidelberg (2003)
15. Gilbert, J.R., Rose, D.J., Edenbrandt, A.: A separator theorem for chordal graphs. SIAM J. on Algebraic and Discrete Methods 5, 306–313 (1984)
16. Gonçalves, D.: Edge partition of planar graphs into two outerplanar graphs. In: 37^{th} Annual ACM Symp. on Theory of Computing (STOC), pp. 504–512. ACM Press, New York (2005)
17. Étude de différents problèmes de partition de graphes, PhD thesis, Université Bordeaux 1, Talence, France, Dec (2006)
18. Kannan, S., Naor, M., Rudich, S.: Implicit representation of graphs. In: 20^{th} Annual ACM Symp. on Theory of Computing (STOC), pp. 334–343. ACM Press, New York (1988)
19. Korman, A., Peleg, D.: Compact separator decompositions and routing in dynamics trees. In: 34^{th} Int'l Colloquium on Automata, Languages and Programming (ICALP). LNCS, vol. 4596, Springer, Heidelberg (2007)
20. Norine, S., Robertson, N., Thomas, R., Wollan, P.: Proper minor-closed families are small. J. of Combinatorial Theory, Series B 96(5), 754–757 (2006)
21. Pagh, R.: Low redundancy in static dictionaries with constant query time. SIAM J. on Computing 31(2), 353–363 (2001)
22. Reed, B.: Finding approximate separators and computing treewidth quickly. In: 24^{th} Annual ACM Symp. on Theory of Comp (STOC), pp. 221–228. ACM Press, New York (1992)
23. Robertson, N., Seymour, P.D.: Graph minors. III. planar tree-width. J. of Combinatorial Theory, Series B 36, 49–64 (1984)
24. Schnyder, W.: Embedding planar graphs on the grid. In: 1^{st} Symp. on Discrete Algorithms (SODA), pp. 138–148. ACM Press, New York (1990)
25. Thorup, M., Zwick, U.: Compact routing schemes. In: 13^{th} Annual ACM Symp. on Parallel Algorithms and Architectures (SPAA), pp. 1–10. ACM Press, New York (2001)