Labeling schemes for flow and connectivity

(Extended abstract)

Michal Katz * Nir A. Katz * Amos Korman † David Peleg † ‡

Abstract

This paper studies labeling schemes for flow and connectivity functions. A flow labeling scheme using $O(\log n \cdot \log \hat{\omega})$-bit labels is presented for general n-vertex graphs with maximum (integral) capacity $\hat{\omega}$. This is shown to be asymptotically optimal. For edge-connectivity, this yields a tight bound of $\Theta(\log^2 n)$ bits. A k-vertex connectivity labeling scheme is then given for general n-vertex graphs using at most $3 \log n$ bits for $k = 2$, $5 \log n$ bits for $k = 3$ and $2^k \log n$ bits for $k > 3$. Finally, a lower bound of $\Omega(k \log n)$ is established for k-vertex connectivity on n-vertex graphs where k is polylogarithmic in n.

1 Introduction

1.1 Problem and motivation.

Network representations play an extensive role in the areas of distributed computing and communication networks. Their goal is to cheaply store useful information about the network and make it readily and conveniently accessible. This is particularly significant when the network is large and geographically dispersed, and information about its structure must be accessed from various local points in it.

The current paper deals with a network representation method based on assigning *informative labels* to the vertices of the network. In most traditional network representations, the names or identifiers given to the vertices contain no useful information, and they serve only as pointers to entries in the data structure, which forms a *global* representation of the network. In contrast, the labeling schemes studied here involve using more informative and localized labels for the network vertices. The idea is to associate with each vertex a label selected in a such way, that will allow us to infer information about any two vertices *directly* from their labels, without using *any* additional information sources. Hence in essence, this method bases the entire representation on the set of labels alone.

Obviously, labels of unrestricted size can be used to encode any desired information, including in particular the entire graph structure. Our focus is thus on informative labeling schemes using relatively *short* labels (say, of length polylogarithmic in n). Labeling schemes of this type were developed in the past for different graph families and for a variety information types, including vertex adjacency [3, 4, 11, 5], distance [16, 15, 10, 7, 13, 8], tree ancestry [1, 12, 2], and various other tree functions, such as center, least common ancestor, separation level or Steiner weight of a given subset of vertices [17]. See the survey [9].

The current paper studies informative labeling schemes for flow and connectivity problems. These types of information are useful in the decision making process required for various reservation-based routing and connection establishment mechanisms in communication networks, in which it is desirable to have accurate information about the potential capacity of available routes between any two given endpoints.

1.2 Labeling schemes.

Let us first formalize the notion of informative labeling schemes. A *vertex-labeling* of the graph G is a function L assigning a label $L(u)$ to each vertex u of G. A labeling scheme is composed of two major components. The first is a *marker* algorithm $\mathcal{M}$, which given a graph G, selects a label assignment $L = \mathcal{M}(G)$ for G. The second component is a *decoder* algorithm $\mathcal{D}$, which given a set of labels $\hat{L} = \{L_1, \ldots, L_k\}$, returns a value $\mathcal{D}(\hat{L})$. The time complexity of the decoder is required to be polynomial in its input size.

Let f be a function defined on sets of vertices in a graph. Given a family $\mathcal{G}$ of weighted graphs, an f *labeling scheme* for $\mathcal{G}$ is a marker-decoder pair $\langle \mathcal{M}_f, \mathcal{D}_f \rangle$ with the following property. Consider any graph $G \in \mathcal{G}$, and let $L = \mathcal{M}_f(G)$ be the vertex labeling assigned by the marker $\mathcal{M}_f$ to G. Then for any set of vertices $W = \{v_1, \ldots, v_k\}$ in G, the value returned by the decoder $\mathcal{D}_f$ on the set of labels $\hat{L}(W) = \{L(v) \mid v \in W\}$

*Department of Mathematics, Bar Ilan University, Ramat Gan, 52900, Israel. E-mail: nir_michal@hotmail.com.

†Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100 Israel. E-mail: {pandit,peleg}@wisdom.weizmann.ac.il.

‡Supported in part by grants from the Israel Science Foundation and the Israel Ministry of Science and Art.

satisfies $\mathcal{D}_f(\hat{L}(W)) = f(W)$.

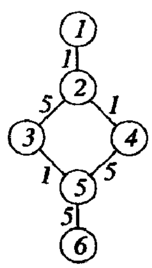
It is important to note that the decoder $\mathcal{D}_f$, responsible of the f-computation, is independent of G or of the number of vertices in it. Thus $\mathcal{D}_f$ can be viewed as a method for computing f-values in a "distributed" fashion, given any set of labels and knowing that the graph belongs to some specific family $\mathcal{G}$. In particular, it must be possible to define $\mathcal{D}_f$ as a constant size algorithm. In contrast, the labels contain some information that can be precomputed by considering the whole graph structure.

For a labeling L for the graph $G = \langle V, E\rangle$, let $|L(u)|$ denote the number of bits in the (binary) string $L(u)$. Given a graph G and a marker algorithm $\mathcal{M}$ which assigns the labeling L to G, denote $\mathcal{L}_{\mathcal{M}}(G) = \max_{u\in V}|L(u)|$. For a finite graph family $\mathcal{G}$, set $\mathcal{L}_{\mathcal{M}}(\mathcal{G}) = \max\{\mathcal{L}_{\mathcal{M}}(G) \mid G \in \mathcal{G}\}$. Finally, given a function f and a graph family $\mathcal{G}$, let

$$\mathcal{L}(f,\mathcal{G}) = \min\{\mathcal{L}_{\mathcal{M}}(\mathcal{G}) \mid \exists\mathcal{D},\ \langle\mathcal{M},\mathcal{D}\rangle \text{ is an } f \text{ labeling scheme for } \mathcal{G}\}.$$

1.3 Flow and connectivity.

In the current paper we focus on flow and connectivity labeling schemes. Let G be a weighted undirected graph $G = \langle V, E, \omega\rangle$, where for every edge $e \in E$, the weight $\omega(e)$ represents the *capacity* of the edge. For two vertices $u, v \in V$, the *maximum flow* possible between them (in either direction), denoted $\mathtt{flow}(u, v)$, is defined as follows. The maximum flow in a path $p = (e_1, e_2, \ldots, e_m)$ is the maximum value that does not exceed the capacity of any edge in the path, i.e., $\mathtt{flow}(p) = \min_{1\le i\le m}\{\omega(e_i)\}$. A set of paths P in G is *edge-disjoint* if each edge $e \in E$ appears in no more than one path $p \in P$. The maximum flow in a set P of edge-disjoint paths is $\mathtt{flow}(P) = \Sigma_{p\in P}\mathtt{flow}(p)$. Let $\mathcal{P}_{u,v}$ be the collection of all sets P of edge-disjoint paths between u and v. Then $\mathtt{flow}(u, v) = \max_{P\in\mathcal{P}_{u,v}}\{\mathtt{flow}(P)\}$. See Figure 1.



	1	2	3	4	5	6
1	∞	1	1	1	1	1
2		∞	6	2	2	2
3			∞	2	2	2
4				∞	6	5
5					∞	5
6						∞

Figure 1: A capacitated graph G, and the (symmetric) flow between its vertices.

As a special case of the flow function, the *edge-connectivity* $\mathtt{e\text{-}conn}(u, w)$ of two vertices u and w in a graph can be given an alternative definition as the maximum flow between the two vertices assuming each edge is assigned one capacity unit.

A set of paths P connecting the vertices u and w in G is *vertex-disjoint* if each vertex except u and w appears in at most one path $p \in P$. The *vertex-connectivity* $\mathtt{v\text{-}conn}(u, w)$ of two vertices u and w in an unweighted graph equals the cardinality of the largest set P of vertex-disjoint paths connecting them. By Menger's theorem (cf. [6]), for nonadjacent u and w, $\mathtt{v\text{-}conn}(u, w)$ equals the minimum number of vertices in $G \setminus \{u, w\}$ whose removal from G disconnects u from w. (When a vertex is removed, all its incident edges are removed as well.)

1.4 Our results.

In this paper we present a number of results concerning labeling schemes for maximum flow, edge-connectivity and vertex-connectivity. In Section 2 we present a flow labeling scheme for general graphs, with label size $O(\log n \cdot \log\hat{\omega})$ over n-vertex graphs with maximum (integral) capacity $\hat{\omega}$. The scheme relies on the fact that the relation "x and y admit a flow of k or more" is an equivalence relation. In the full paper [14] we also establish the optimality of our flow labeling scheme by proving a tight lower bound of $\Omega(\log n \cdot \log\hat{\omega})$ on the required label size for flow labeling schemes on the class of n-vertex trees with maximum capacity $\hat{\omega}$. For edge-connectivity, this yields a tight bound of $\Theta(\log^2 n)$.

In comparison, vertex connectivity seems to require a more involved labeling scheme whose label size depends on the connectivity parameter k. In Section 3 we present a k-vertex-connectivity labeling scheme for general n-vertex graphs. The label sizes we achieve are $\log n$ for $k = 1$, $3\log n$ for $k = 2$, $5\log n$ for $k = 3$ and $2^k\log n$ for $k > 3$. In Section 4 we present a lower bound of $\Omega(k\log n)$ for the required label size for k-vertex connectivity on general n-vertex graphs, where k is polylogarithmic in n.

2 Flow labeling schemes for general graphs

In this section we consider the family $\mathcal{G}(n,\hat{\omega})$ of undirected capacitated connected n-vertex graphs with maximum (integral) capacity $\hat{\omega}$, and present a flow labeling scheme for this family with label size $O(\log n \cdot \log\hat{\omega})$. Given a graph $G = \langle V, E, \omega\rangle$ in this family and an integer $1 \le k \le \hat{\omega}$, let us define the following relation:

$$R_k = \{(x, y) \mid x, y \in V,\ \ \mathtt{flow}(x, y) \ge k\}.$$

We make use of the following easy to prove fact. (Throughout, some proofs are omitted.)

LEMMA 2.1. *The relation R_k is an equivalence relation.*

For every $k \geq 1$, the relation R_k induces a collection of equivalence classes on V, $\mathcal{C}_k = \{C_k^1, \ldots, C_k^{m_k}\}$, such that $C_k^i \cap C_k^j = \emptyset$ and $\bigcup_i C_k^i = V$. Note that for $k < k'$, the relation $R_{k'}$ is a *refinement* of R_k, namely, for every class $C_{k'}^i$ there is a class C_k^j such that $C_{k'}^i \subseteq C_k^j$.

Given G, let us construct a tree T_G corresponding to its equivalence relations. The k'th level of T corresponds to the relation R_k, i.e., it has m_k nodes, marked by the classes $C_k^1, \ldots, C_k^{m_k}$. In particular, the root of T is marked by the unique equivalence class of R_1, which is V. The tree is truncated at a node once the equivalence class associated with it is a singleton. For every vertex $v \in G$, denote by $t(v)$ the leaf in T_G associated with the singleton set $\{v\}$. Figure 2 describes the tree T_G corresponding to the flow equivalence classes for the graph G of Figure 1.

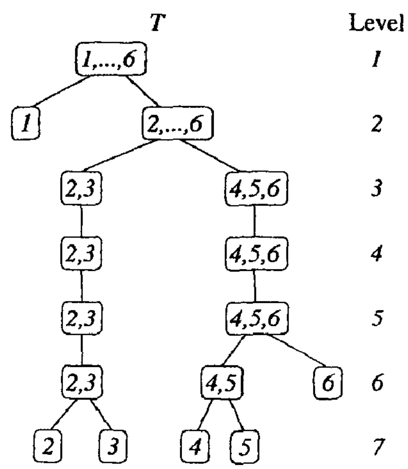


Figure 2: The tree T_G corresponding to the graph G of Figure 1.

For two nodes x, y in a tree T rooted at r, define the *separation level* of x and y, denoted $\texttt{SepLevel}_T(x, y)$, as the depth of $z = lca(x, y)$, the least common ancestor of x and y. I.e., $\texttt{SepLevel}_T(x, y) = dist_T(z, r)$, the distance of z from the root. As an immediate consequence of the construction, we have the following connection.

LEMMA 2.2. *For every two vertices* $v, w \in V$, $\texttt{flow}_G(v, w) = \texttt{SepLevel}_T(t(v), t(w)) + 1$.

It is proven in [17] that for the class $\mathcal{T}(n)$ of n-node unweighted trees, there exists a SepLevel labeling scheme with $O(\log^2 n)$-bit labels. (This is also shown to be optimal, in the sense that any such scheme must label some node of some n-node unweighted tree with an $\Omega(\log^2 n)$-bit label.)

Observe that if the maximum capacity of any edge in the n-vertex graph G is $\hat{\omega}$, then the depth of the tree T_G cannot exceed $\hat{\omega}$ levels, and it may have at most n nodes per level, hence the total number of nodes in T_G is $O(n\hat{\omega})$. We immediately have that $\mathcal{L}(\texttt{flow}, \mathcal{G}(n, \hat{\omega})) = O(\log^2(n\hat{\omega}))$.

A more careful design of the tree T_G can improve the bound on the label size. This is achieved by canceling all nodes of degree 2 in the tree T_G, and adding appropriate edge weights. Specifically, a subpath $(v_0, v_1, \ldots, v_k)$ in T_G such that $k \geq 2$, v_0 and v_k have degree 3 or higher, and $v_1, \ldots, v_{k-1}$ have degree 2 (with $v_1, \ldots, v_k$ all marked by the same set C) is compacted into a single edge (v_0, v_k) with weight k, eliminating the nodes $v_1, \ldots, v_{k-1}$, and leaving the sets marking the remaining nodes unchanged. Let $\tilde{T}_G$ denote the resulting compacted tree. Figure 3 describes the tree $\tilde{T}_G$ corresponding to the tree T_G of Figure 2.

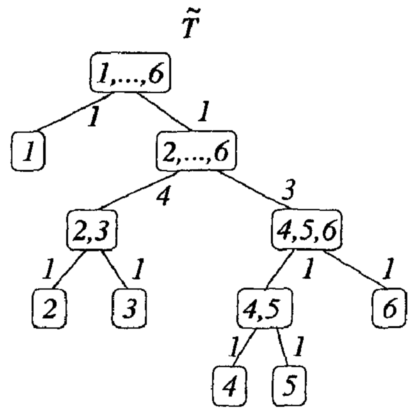


Figure 3: The compacted tree $\tilde{T}_G$ corresponding to the tree T_G of Figure 2.

The notion of separation level can be extended to weighted rooted trees in the natural way, by defining $\texttt{SepLevel}_T(x, y)$ as the weighted depth of $z = lca(x, y)$, i.e., its weighted distance from the root. The upper and lower bounds presented in [17] regarding SepLevel labeling schemes for unweighted trees can also be extended in a straightforward manner to weighted trees, yielding SepLevel labeling schemes for the class $\mathcal{T}(\tilde{n}, \tilde{\omega})$ of weighted $\tilde{n}$-node trees with maximum weight $\tilde{\omega}$ using $O(\log \tilde{n} \log \tilde{\omega} + \log^2 \tilde{n})$-bit labels.

It is also easy to verify that for two nodes x, y in G, the separation level of the leaves $t(x)$ and $t(y)$ associated with x and y in the tree $\tilde{T}_G$ is still related to the flow between the two vertices as characterized in Lemma 2.2.

Finally, note that as $\tilde{T}_G$ has exactly n leaves, and every non-leaf node in it has at least two children, the total number of nodes in $\tilde{T}_G$ is $\tilde{n} \leq 2n - 1$. Moreover, the maximum edge weight in $\tilde{T}_G$ is $\tilde{\omega} \leq \hat{\omega}$.

Combining the above observations, we have the following.

THEOREM 2.1. $\mathcal{L}(\texttt{flow}, \mathcal{G}(n,\hat{\omega})) = O(\log n \cdot \log\hat{\omega} + \log^2 n)$. ∎

The above theorem immediately yields the following upper bound for edge-connectivity. Let $\mathcal{G}(n)$ denote the class of n-vertex unweighted graphs.

COROLLARY 2.1. $\mathcal{L}(\text{e-conn}, \mathcal{G}(n)) = O(\log^2 n)$. ∎

Finally we give a lower bound of $\Omega(\log n \cdot \log\hat{\omega})$ on the label size for flow on the class $\mathcal{T}(n,\hat{\omega})$ of n-vertex trees with maximum edge capacity $\hat{\omega}$ (which is assumed to be integral). The proof idea is based on a modification of the lower bound proof of [10] for distance labeling schemes, and is omitted from this extended abstract (see [14]).

THEOREM 2.2. *For* $\hat{\omega} > \log(n+1) - 1$, $\mathcal{L}(\texttt{flow}, \mathcal{T}(n,\hat{\omega})) = \Omega(\log n \log\hat{\omega})$. ∎

COROLLARY 2.2. $\mathcal{L}(\text{e-conn}, \mathcal{G}(n)) = \Theta(\log^2 n)$. ∎

3 Vertex-connectivity labeling schemes for general graphs

In this section we turn to k-vertex-connectivity, and present a labeling scheme for general n-vertex graphs. The label sizes we achieve are $\log n$ for $k=1$, $3\log n$ for $k=2$, $5\log n$ for $k=3$ and $2^k \log n$ for $k>3$.

3.1 Preliminaries.

We start with some preliminary definitions. In an undirected graph G, two vertices are called *k-connected* if there exist at least k vertex-disjoint paths between them. A set $S \subseteq V$ *separates* u from v in $G = \langle V,E\rangle$ if u and v are not connected in the vertex induced subgraph $G \setminus S$.

THEOREM 3.1. **[Menger] (cf. [6])** *In an undirected graph G, two nonadjacent vertices u and v are k-connected iff no set $S \subset G\setminus\{u,v\}$ of $k-1$ vertices can separate u from v in G.*

The *k-connectivity graph* of $G = \langle V,E\rangle$ is $C_k(G) = \langle V,E'\rangle$, where $(u,v) \in E'$ iff u and v are k-connected in G. A graph G is *closed under k-connectivity* if it has the property that if u and v are k-connected in G then they are neighbors in G. Let $\mathcal{C}(k)$ be the family of all graphs G which are closed under k-connectivity.

OBSERVATION 3.1. *1. If $G \in \mathcal{C}(k)$ then each connected component of G belongs to $\mathcal{C}(k)$.*

2. If $G = H \cup F$ where $H,F \in \mathcal{C}(k)$ are vertex-disjoint subgraphs of G, then $G \in \mathcal{C}(k)$.

A graph G is called *k-orientable* if there exists an orientation of the edges such that the out-degree of each vertex is bounded above by k. The class of k-orientable graphs is denoted $\mathcal{J}_{or}(k)$.

OBSERVATION 3.2. *If $G = H \cup F$ where $H,F \in \mathcal{J}_{or}(k)$ are vertex-disjoint subgraphs of G, then $G \in \mathcal{J}_{or}(k)$.*

LEMMA 3.1. *Let $G' = \langle V,E'\rangle$ where $E' = E \cup \{(u,v)\}$ for some pair of k-connected vertices u and v. Then G and G' have the same k-connectivity graph, i.e., $C_k(G) = C_k(G')$.*

Proof: Use induction on k. For k=1 the Lemma is obvious. Assume the Lemma is true for $k-1$. It suffices to show that if two vertices w,w' are not k-connected in G then they are not k-connected in G'. Suppose that w,w' are not k-connected in G. If w,w' are neighbors in G then let $G^- = G\setminus\{u,v\}$. In G^-, w and w' are not $k-1$-connected and since u and v are $k-1$ connected in G^- , by induction hypothesis w and w' are not $k-1$-connected in $G'\setminus\{u,v\}$. This implies that they are not k connected in G' as desired. If w,w' are not neighbors in G then by Menger's theorem there exists a set of vertices $S = \{x_1, x_2, \ldots, x_{k-1}\}$ that separates w from w' in G. We claim that S separates w from w' also in G'. The proof breaks into the following cases.

- **Case 1:** One or more of the x_i's is u or v. Then $G\setminus S = G'\setminus S$.
- **Case 2:** None of the x_i's is u or v. If u and v belong to the same connectivity component of $G\setminus S$ then the connectivity components of $G'\setminus S$ will be the same as the connectivity components of $G\setminus S$, implying that S separates w from w' also in G', which is what we wanted to prove. If u and v belong to different connectivity components of $G\setminus S$ then S separates u from v in G, or in other words, u and v are not k-connected in G, contradicting our assumption. ∎

COROLLARY 3.1. *For every graph G, If u and v are k-connected in $C_k(G)$ then they are neighbors in $C_k(G)$, i.e., $C_k(G) \in \mathcal{C}(k)$.*

Proof: Transform a given graph G into $G+ = G \cup C_k(G)$ by adding the edges of $C_k(G)$ to G, one by one. By induction on the steps of this process using the previous Lemma, we get $C_k(G^+) = C_k(G)$. Therefore if u and v were k-connected in $C_k(G)$ then they are k-connected in G^+ and therefore they are neighbors in $C_k(G^+) = C_k(G)$. ∎

For a connectivity component C of $C_k(G)$, a *left-most BFS* tree for C, denoted $T(C,k)$, is a BFS tree spanning C, constructed in the following way. Take a vertex r from C to be the root of $T(C,k)$. Let $level(r) = 1$. Assume we constructed i levels of $T(C,k)$ and haven't used all vertices of C. Construct the $(i+1)$'st level of $T(C,k)$ as follows. Repeatedly take a vertex v of level i and connect it to all the vertices adjacent to it in $C_k(G)$ that haven't been included so far in the tree construction. For each such new vertex w let $level(w) = i+1$ and let v be w's parent in $T(C,k)$.

When the context is clear we use the notation T instead of $T(C,k)$.

For $T = T(C,k)$, we make the following definitions. Let W_i denote the set of vertices of level i in T, and let $H_i = H_i(C,k) = \langle W_i, E_i \rangle$ be the subgraph of C induced by W_i. For vertices u and v, denote u's parent in T by $p(u)$ and let $lca(u,v)$ be the highest level common ancestor of both u and v in T. Let W'_{i+1} denote the set of vertices of W_{i+1} that neighbor at least k vertices of W_i in $C_k(G)$. Let $F_i = F_i(C,k)$ be the subgraph of C induced by $W_i \cup W'_{i+1}$.

LEMMA 3.2. *1. For $T = T(C,k)$, $H_i \in \mathcal{C}(k-1)$.*

2. For $T = T(C,k)$, $F_i \in \mathcal{C}(k-1)$.

Proof: To prove (1), we show that every two vertices $u, v \in W_i$ that are $(k-1)$-connected in H_i, are neighbors in $C_k(G)$ and therefore in H_i, implying $H_i \in \mathcal{C}(k-1)$. Assume, for contradiction, that u and v are not neighbors in $C_k(G)$. By Corollary 3.1 they are also not k-connected in $C_k(G)$, i.e., there exists a set $S = \{x_1, \ldots, x_{k-1}\}$ that separates them in $C_k(G)$. Let $S' = S \cap W_i$. Since S' separates u from v in H_i and since u and v are $(k-1)$-connected in H_i we get that $|S'| = k-1$ hence $S' = S$, so all the vertices in S must be of level i. But then, S does not separate u from v even in T, which is a subgraph of $C_k(G)$, contradicting our assumption.

Turning to (2), let u and v be $(k-1)$-connected in F_i. As before, it suffices to show that they are neighbors in $C_k(G)$. Assume for contradiction that u and v are not neighbors in $C_k(G)$ therefore they are also not k-connected in $C_k(G)$, i.e., there exists a set $S = \{x_1, \ldots, x_{k-1}\}$ that separates them in $C_k(G)$. Since $S' = S \cap F_i$ separates u from v in F_i and since u and v are $(k-1)$-connected in F_i we get, as before, that all the vertices of S must belong to F_i.

- **Case 1:** Both u and v are of level i. In this case, as before, S does not separate u from v even in T, which is a subgraph of $C_k(G)$, contradicting our assumption.
- **Case 2:** Without loss of generality u is of level i, v is of level $i+1$ and v has at least k neighbors in $C_k(G)$ of level i. In this case, v has at least one neighbor w of level i in $C_k(G) \setminus S$. Since all vertices of S are in F_i, w and u are connected in $C_k(G) \setminus S$ via the edges of T. Altogether we get that u and v are connected in $C_k(G) \setminus S$, contradicting our assumption. ∎

3.2 Overview of the scheme.

We rely on the basic observation that labeling k-connectivity for some graph G is equivalent to labeling adjacencies for $C_k(G)$. By Corollary 3.1, $C_k(G) \in \mathcal{C}(k)$. Therefore, instead of presenting a k-connectivity labeling scheme for general graphs, we present an adjacency labeling scheme for the graphs of $\mathcal{C}(k)$.

The general idea used for labeling adjacencies for some $G \in \mathcal{C}(k)$, especially for $k > 3$, is to decompose G into at most 3 'simpler' graphs. One of these graphs is a k-orientable graph K, and the other two, called G_{even} and G_{odd}, belong to $\mathcal{C}(k-1)$. The labeling algorithm for $G \in \mathcal{C}(k)$ recursively labels subgraphs of G that belong to $\mathcal{C}(t)$ for $t < k$. When we are concerned with labeling some n-vertex graph $G \in \mathcal{C}(k)$ for $k > 1$, the first step in the labeling is to assign each vertex u in G a distinct identity $id(u)$ from 1 to n. This identity will always appear as the last $\log n$ bits of the label $L(G,u)$. Thus, when labeling the subgraphs of G in the recursion we may assume that the id's for the vertices are given.

For graphs $G = \langle V, E \rangle$ and $G_i = \langle V_i, E_i \rangle$, $i > 1$, we say that G can be *decomposed* into the G_i's if $\bigcup_i V_i = V$, $\bigcup_i E_i = E$ and the E_i's are pairwise disjoint.

LEMMA 3.3. *Let $\mathcal{G}, \mathcal{G}_1$ and $\mathcal{G}_2$ be families of graphs such that each $G \in \mathcal{G}$ can be decomposed into $G_1 \in \mathcal{G}_1$ and $G_2 \in \mathcal{G}_2$. If $\mathcal{G}_1$ and $\mathcal{G}_2$ have adjacency labeling schemes of sizes l_1 and l_2 respectively, then $\mathcal{G}$ has adjacency labeling scheme of size $l_1 + l_2$.*

Proof: The general idea in the proof is to use concatenation of the labels of the decomposed graphs. Let $\langle \mathcal{M}_i, \mathcal{D}_i \rangle$ be adjacency labeling schemes for $\mathcal{G}_i$ $(i = 1,2)$. Let us construct an adjacency labeling scheme $\langle \mathcal{M}, \mathcal{D} \rangle$ for $\mathcal{G}$ as follows.

The marker algorithm $\mathcal{M}$ for $\mathcal{G}$: For a given graph $G \in \mathcal{G}$, decompose G into $G_i \in \mathcal{G}_i$ $(i = 1,2)$. Let $L_i = \mathcal{M}_i(G_i)$ for i=1,2. We construct $L = \mathcal{M}(G)$ as follows. For a vertex u in G, let $L(u) = \langle L_1(u), L_2(u) \rangle$ where the first l_1 bits of the label $L(u)$ consist of $L_1(u)$ and the next l_2 bits give $L_2(u)$. Altogether we use $l_1 + l_2$ bits.

The decoder algorithm $\mathcal{D}$ for $\mathcal{G}$: Let G, G_1 and G_2 be as before. Given the two labels $L(u) = \langle L_1(u), L_2(u) \rangle$ and $L(v) = \langle L_1(v), L_2(v) \rangle$ let $\mathcal{D}(L(u), L(v)) = \mathcal{D}_1(L_1(u), L_1(v)) \vee \mathcal{D}_2(L_2(u), L_2(v))$.

Since G was decomposed into G_1, G_2 the vertices u and v are neighbors in G iff they are neighbors in G_1 or in G_2, hence the decoding algorithm is correct. ▮

COROLLARY 3.2. *Let $\mathcal{G}, \mathcal{G}_1, ..., \mathcal{G}_m$ be families of graph such that each $G \in \mathcal{G}$ can be decomposed into $G_1, ..., G_m$ were $G_i \in \mathcal{G}_i$ for $i = 1$ to m. If the $\mathcal{G}_i$'s have adjacency labeling schemes of sizes l_i respectively, then $\mathcal{G}$ has an adjacency labeling scheme of size $\sum l_i$.*

LEMMA 3.4. *Let $\mathcal{J}_n(k)$ be the family of n-vertex graphs in $\mathcal{J}_{or}(k)$. Assuming id's are given,*
$\mathcal{L}(\texttt{adjacency}, \mathcal{J}_n(k)) \leq k \log n$.

Proof: Suppose $G \in \mathcal{J}_n(k)$ then G is a k-orientable graph with n vertices. Hence there exists an orientation to the edges of G such that the out-degree of each vertex is bounded above by k. In this orientation, for each u there exist at most k outgoing edges, say (u, v_1),(u, v_2),..., (u, v_t), for $t \leq k$.
The marker algorithm $\mathcal{M}$ for $\mathcal{J}_n(k)$: Label u by $L(u) = \langle id(v_1), id(v_2), ..., id(v_t) \rangle$, i.e., use the first $\log n$ bits to write $id(v_1)$, the second $\log n$ bits to write $id(v_2)$, etc. Hence, for every u's, the size of $L(u)$ is at most $k \log n$ bits.
The decoder algorithm $\mathcal{D}$ for $\mathcal{J}_n(k)$: Given $L(u)$ and $L(v)$, check whether u's id appears in $L(v)$, by inspecting each block of $\log n$ bits in $L(v)$ separately. Analogously, check if v's id appears in $L(u)$.
As u and v are neighbors in G iff one of the two cases applies, the decoding algorithm is correct. ▮

To illustrate the approach, we precede the treatment of the general case with a discussion of the cases $k = 1, 2, 3$, for which slightly better schemes are available. The simple case of $k = 1$ is handled in Section 3.2.1. For $k = 2$ we show in Section 3.2.2 that a connected graph $G \in \mathcal{C}(2)$ can be decomposed into a tree and disjoint graphs in $\mathcal{C}(1)$. Graphs in $\mathcal{C}(1)$ are collections of cliques. It follows that each $G \in \mathcal{C}(2)$ can be decomposed into a forest (which is a 1-orientable graph) and a graph made of disjoint cliques. For $k = 3$ we show in Section 3.2.3 that a connected graph $G \in \mathcal{C}(3)$ can be decomposed into a graph in $\mathcal{C}(2)$ and a 2-orientable graph.

3.2.1 A 1-connectivity labeling scheme.

Let us give a labeling scheme for 1-connectivity for $\mathcal{G}_n$, the family of all n-vertex graphs.
The marker algorithm $\mathcal{M}$ for $\mathcal{G}_n$: Fix $G = \langle V, E \rangle \in \mathcal{G}_n$. To each connected component C of G assign a distinct identity $id(C)$ from the range $\{1, \ldots, n\}$. For a vertex $u \in V$, let C_u be the connected component of G that u belongs to. The marker algorithm sets $L(u) = id(C_u)$.
The Decoder $\mathcal{D}$ for $\mathcal{G}_n$: Let $D(L(u), L(v)) = 1$ iff $L(u) = L(v)$.

Clearly u and v are 1-connected in G iff they are in the same connected component, hence the decoder's response is correct. The size of the label is bounded above by $\log n$.

THEOREM 3.2. *$\mathcal{L}(1-\texttt{v-conn}, \mathcal{G}_n) \leq \log n$.* ▮

3.2.2 A 2-connectivity labeling scheme.

As explained earlier, labeling 2-connectivity for a family of graphs $\mathcal{G}$ is equivalent to labeling adjacencies for the family $\{C_2(G) : G \in \mathcal{G}\} \subseteq \mathcal{C}(2)$. In this section we present an efficient adjacency labeling scheme for $\mathcal{C}(2)$.

Consider a graph $G \in \mathcal{C}(2)$ and let $C_1, ..., C_m$ be its connected components. By Observation 3.1(1) $C_i \in \mathcal{C}(2)$ for every i. Fix i and let $T = T(C_i, 2)$.

CLAIM 3.1. *The only neighbor of u in G which has a strictly lower level than u in T is $p(u)$.*

Proof: Suppose, for contradiction, that there exist neighbors v and w such that $level(w) > level(v)$ but v is not $p(w)$ in T. In this case, w and $z = lca(v, w)$ are 2-connected in $T \cup \{(v, w)\}$, which is a subgraph of G. Since $G \in \mathcal{C}(2)$, v must be a neighbor of w. Since $level(w) < level(u) - 1$ we get a contradiction to the way T was constructed. ▮

CLAIM 3.2. *$G \in \mathcal{C}(2)$ can be decomposed into a forest F and a graph H of disjoint cliques.*

Proof: Fix a connected component C_i of G and let $T = T(C_i, 2)$. Since, by Lemma 3.2(1), each H_j^i, subgraph of C_i induced by level j of T, is in $\mathcal{C}(1)$, it follows that H_j^i is a collection of disjoint cliques. Hence G can be decomposed into a forest F and a graph of disjoint cliques, H composed of the collection of all the H_j^i from all i's and j's. ▮

Let $\mathcal{C}_n(2)$ be the family n-vertex graphs in $\mathcal{C}(2)$. Let us now give an adjacency labeling scheme for the graphs of $\mathcal{C}_n(2)$.
The marker algorithm $\mathcal{M}$ for $\mathcal{C}_n(2)$: Decompose G into F and H as in Claim 3.2. Fix a vertex u of G. Let $p(u)$ be u's parent in F. To each clique C in H give a distinct identity from the range $\{1, \ldots, n\}$, $id(C)$. Let $C(u)$ be the clique in H that contains u.

The marker algorithm for G assigns $L(u) = \langle id(c(u), id(p(u)), id(u) \rangle$. As before we use the first $\log n$ bits for $id(c(u))$ the second $\log n$ bits for $id(p(u))$ etc. The label size is bounded above by $3 \log n$.
The Decoder $\mathcal{D}$ for $\mathcal{C}_n(2)$: Given $L(u)$ and $L(v)$ we compare $id(p(u))$ with $id(v)$ and $id(p(v))$ with $id(u)$ to

check whether one is the parent of the other in the forest F. We also check if $id(C(u)) = id(C(v))$ to see whether u and v are neighbors in H. We do this by looking at the corresponding bits in the label, for example, $id(p(u))$ is written in the second block of $\log n$ bits of $L(u)$. Let $\mathcal{D}(L(u), L(v)) = 1$ iff either $id(C(u)) = id(C(v))$, $id(p(u)) = id(v)$ or $id(p(v)) = id(u)$.

Clearly, u and v are neighbors in G iff they are neighbors in F or in H, hence the decoder's response is correct. We get the following.

THEOREM 3.3. *Let $\mathcal{G}_n$ be the family of n-vertex graphs then $\mathcal{L}(2-\text{v-conn}, \mathcal{G}_n) \leq 3\log n$ bits.*

3.2.3 A 3-connectivity labeling scheme.

Again, labeling 3-connectivity for a family $\mathcal{G}$ is equivalent to labeling adjacencies for the family $\{C_3(G) : G \in \mathcal{G}\} \subseteq \mathcal{C}(3)$. In this section we show how to label adjacencies for $\mathcal{C}(3)$.

Consider a graph $G \in \mathcal{C}(3)$, and let $C_1, \ldots, C_m$ be its connected components. By observation 3.1(1), $C_i \in \mathcal{C}(3)$ for all i. Fix i and let $T = T(C_i, 3)$.

LEMMA 3.5. *Each vertex u has at most one neighbor of G which has a strictly lower level than u in T apart from $p(u)$.*

Proof: Assume, for contradiction, that there exist a vertex u with two neighbors in G, v and w, both with a strictly lower level than u and both different from $p(u)$. In this case, u must be 3-connected in G to either $lca(u,v)$, $lca(u,w)$ or $lca(v,w)$. However, the levels of $lca(u,v)$, $lca(u,w)$, $lca(v,w)$ are all smaller than $level(u) - 1$, and since $G \in \mathcal{C}(3)$, u is adjacent to one of them, contradicting the way T was constructed. (See Fig. 4.) ▮

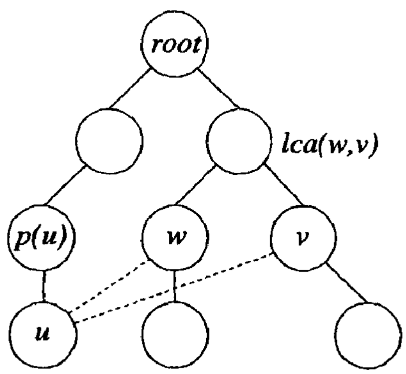


Figure 4: An illustration to the contradiction in the proof of Lemma 3.5.

LEMMA 3.6. *Each $G \in \mathcal{C}(3)$ can be decomposed into a graph $H \in \mathcal{C}(2)$ and a 2-orientable graph.*

Proof: First, it suffices to show the lemma for connected graphs $C \in \mathcal{C}(3)$, since by Observations 3.1(2) and 3.2 $\mathcal{C}(2)$ and $\mathcal{J}_{or}(2)$ are closed under vertex-disjoint unions. Consider a connected graph $C \in \mathcal{C}(3)$ and let $T = T(C, 3)$. By Lemma 3.2(1), each subgraph H_j of C induced by the vertices of level j in T, is in $\mathcal{C}(2)$. All the subgraphs H_j are vertex-disjoint, hence by letting H be the union of all the H_j, we get $H \in \mathcal{C}(2)$. Let U be the graph C after deleting the edges of H. By Lemma 3.5, each vertex u of U has at most 2 neighbors of a strictly lower level (one of which is u's parent in T). Hence directing the edges of U from higher level vertices to lower level vertices, each u has out-degree at most 2, i.e., U is 2-orientable. ▮

By Lemmas 3.3 and 3.4 and from Theorem 3.3 we get the following theorem.

THEOREM 3.4. *Let $\mathcal{G}_n$ be the family of n-vertex graphs, then $\mathcal{L}(3-\text{v-conn}, \mathcal{G}_n) \leq 5\log n$ bits.*

3.3 A k-connectivity labeling scheme.

Finally, labeling k-connectivity for a family $\mathcal{G}$ is equivalent to labeling adjacencies for the family $\{C_k(G) : G \in \mathcal{G}\} \subseteq \mathcal{C}(k)$. In this section we show how to label adjacencies for $\mathcal{C}(k)$.

Consider a graph $G \in \mathcal{C}(k)$, and let $C_1, \ldots, C_m$ be its connected components. By observation 3.1(1), $C_i \in \mathcal{C}(k)$ for all i. Fix i and let $T = T(C_i, k)$.

LEMMA 3.7. *Each $G \in \mathcal{C}(k)$ can be decomposed into two graphs in $\mathcal{C}(k-1)$ and a $(k-1)$-orientable graph.*

Proof: Again, it suffices to prove the lemma for connected graphs $C \in \mathcal{C}(k)$ since by Observations 3.1(2) and 3.2 both $\mathcal{C}(k-1)$ and $\mathcal{J}_{or}(k)$ are closed under vertex-disjoint unions. Consider a connected graph $C \in \mathcal{C}(k)$ and let $T = T(C, k)$.

All the F_i's for odd i's are vertex-disjoint, and $F_i \in \mathcal{C}(k-1)$ for all i's by Lemma 3.2(2). Therefore, by letting G_{odd} be the union of all the F_i's for odd i's, we get $G_{odd} \in \mathcal{C}(k-1)$. For the same reasoning, by letting G_{even} be the union of all the F_i's for even i's, we get $G_{even} \in \mathcal{C}(k-1)$.

Let K be the graph C after omitting the edges of G_{odd} and G_{even} (or equivalently, omitting all edges of all the F_i's). The proof is completed once we show that K is $(k-1)$-orientable. Since all edges (u, v) of C such that $level(u) = level(v) = i$ for some i are in F_i for the appropriate i, if (u, v) is an edge of K then $level(u) \neq level(v)$. By the way T was constructed, the difference between the levels is 1.

Let us direct the edges of K from higher level vertices to lower level vertices. Assume, for contradiction, that for some u and some i, $level(u) = i + 1$ and the out-degree of u in K is at least k. Then u must have at least k neighbors of level i in C, in which case all edges (u,v) for v such that $level(v) = i$ appear in F_i and therefore not in K. Therefore, the out-degree of u in K is 0, contradicting our assumption. ∎

Before stating and proving the next theorem, let us remark that we can get a weaker upper bound of $3^k \log n$ label size for $\mathcal{L}(\texttt{adjacency}, \mathcal{C}_n(k))$ in the following way. Use induction on k. For $k = 1,2,3$ our remark holds. For $k > 3$ fix k and assume that the remark holds for $k-1$. The remark for k follows from Lemmas 3.4 and 3.7 and Corollary 3.2.

To prove the next theorem, we show that instead of concatenating u's labels in the three decomposed graphs (G_{odd}, G_{even}, K), it suffices to give u its label in only two of the three decomposed graphs. This yields the desired $2^k \log n$ bits bound on $\mathcal{L}(\texttt{adjacency}, \mathcal{C}_n(k))$.

THEOREM 3.5. *Let $\mathcal{C}_n(k)$ be the family of n-vertex graphs in $\mathcal{C}(k)$, then*

$$\mathcal{L}(\texttt{adjacency}, \mathcal{C}_n(k)) \leq 2^k \log n.$$

Proof: Use induction on k. For k=1, 2 or 3 the theorem holds as seen in Theorems 3.2, 3.3 and 3.4. For $k > 3$, fix k and assume that the Theorem holds for $k-1$. Consider a graph $G \in \mathcal{C}_n(k)$. For a vertex u in G, let C be its connected component in G, let $T = T(C,k)$ and let $i = level(u)$. Let us now give a labeling scheme for adjacency on $G \in \mathcal{C}(k)$.

The marker algorithm $\mathcal{M}_k$ for $\mathcal{C}(k)$: For $t \leq k$, $G \in \mathcal{C}_n(t)$ and u, a vertex of G, denote the adjacency labeling on G by $L_t(G)$ and u's label by $L_t(G,u)$. Let $G \in \mathcal{C}_n(k)$ and let u be a vertex in G. we define State(u) according to the following three cases:

- **Case 1:** u participates in both G_{odd} and G_{even}. Let State(u) = Dual.

  Note that in this case the out-degree of u in K is 0. The marker algorithm assigns to u the label $L_k(G,u) = \langle L_{k-1}(G_{odd},u), L_{k-1}(G_{even},u)\rangle$ where the first $2^{k-1}\log n$ bits are reserved for $L_{k-1}(G_{odd},u)$ and the last $2^{k-1}\log n$ bits are reserved for $L_{k-1}(G_{even},u)$.

- **Case 2:** u doesn't participate in G_{odd}, i.e., u participates only in G_{even} and in K. Let State(u) = Even. Let $L_k(G,u) = \langle 0^{k\log n}, 10, L(u,K), 00...000, L_{k-1}(G_{even},u)\rangle$ where the two bits in the second field, 10, indicate that State(u) = Even. the next $k\log n$ bits are reserved for $L(u,K)$ and the last $2^{k-1}\log n$ bits are reserved for $L_{k-1}(G_{even},u)$.

- **Case 3:** u doesn't participate in G_{even}, i.e., u participates only in G_{odd} and in K. Let State(u) = Odd. Let $L_k(G,u) = \langle 0^{k\log n}, 11, L(u,K), 00...00, L_{K-1}(G_{odd},u)\rangle$ where the two bits in the second field, 11, indicate that State(u) = Odd, the next $k\log n$ bits are reserved for $L(u,K)$ and the last $2^{k-1}\log n$ bits are reserved for $L_{K-1}(G_{odd},u)$.

By the definition of K, it is clear that the out-degree of some u in K is higher than 0 iff State(u) = Even or Odd.

The Decoder $\mathcal{D}_k$ for $\mathcal{C}(k)$: For $t \leq k$ denote the decoder for $\mathcal{C}(t)$ by $\mathcal{D}_t$. Denote the decoder for $\mathcal{J}_{or}(k)$ (from Lemma 3.4) by $\mathcal{D}_{or}$. Given $L_k(G,u)$ and $L_k(G,v)$ we will first want to know the states of u and v. Take for example $L_k(G,u)$. For $k > 3$, the first $k\log n$ bits are 0 iff State(u) $\neq$ Dual. So by looking at the first $k\log n+2$ bits of $L_k(G,u)$ and $L_k(G,v)$ we know the states of u and v. Consider the following cases:

- **Case a:** State(u) = State(v) = Dual: Then $\mathcal{D}_k$ for G uses $\mathcal{D}_{k-1}$ on G_{even} and G_{odd} as follows.

$$\mathcal{D}_k(L_k(G,u), L_k(G,v)) =$$
$$(\mathcal{D}_{k-1}(L_{k-1}(G_{odd},u), L_{k-1}(G_{odd},v))) \vee$$
$$(\mathcal{D}_{k-1}(L_{k-1}(G_{even},u), L_{k-1}(G_{even},v)))$$

- **Case b:** State(u) = State(v) = Even: Then $\mathcal{D}_k$ for G uses $\mathcal{D}_{k-1}$ on G_{even} and $\mathcal{D}_{or}$ for K as follows.

$$\mathcal{D}_k(L_k(G,u), L_k(G,v)) =$$
$$\mathcal{D}_{or}(L(u,K), L(v,K)) \vee$$
$$\mathcal{D}_{k-1}(L_{k-1}(G_{even},u), L_{k-1}(G_{even},v))$$

- **Case c:** State(u) = State(v) = Odd: Then $\mathcal{D}_k$ for G uses $\mathcal{D}_{k-1}$ on G_{odd} and $\mathcal{D}_{or}$ for K as follows.

$$\mathcal{D}_k(L_k(G,u), L_k(G,v)) =$$
$$\mathcal{D}_{or}(L(u,K), L(v,K)) \vee$$
$$\mathcal{D}_{k-1}(L_{k-1}(G_{odd},u), L_{k-1}(G_{odd},v))$$

- **Case d:** State(u) = Dual, State(v) = Even: Then let $\mathcal{D}_k(L_k(G,u), L_k(G,v)) = 1$ if and only if $\mathcal{D}_{k-1}(L_{k-1}(G_{even},u), L_{k-1}(G_{even},v)) = 1$ or $id(u)$ appears in $L(v,K)$.

- **Case e:** State(u) = Dual, State(v) = Odd: Then let $\mathcal{D}_k(L_k(G,u), L_k(G,v)) = 1$ if and only if $\mathcal{D}_{k-1}(L_{k-1}(G_{odd},u), L_{k-1}(G_{odd},v)) = 1$ or $id(u)$ appears in $L(v,K)$.

- **Case f:** State(u) = Even, State(v) = Odd: Then

$$\mathcal{D}_k(L_k(G,u), L_k(G,v)) = \mathcal{D}_{or}(L(u,K), L(v,K))$$

To prove correctness, use induction on k. If u and v are neighbors of level i then the edge (u,v) appears in F_i and therefore u and v participate both in either G_{odd} or in G_{even} depending on the parity of i. Thus, by comparing the appropriate labels, say $L_{k-1}(G_{odd},u)$ and $L_{k-1}(G_{odd},v)$, we can deduce that u and v are indeed neighbors by the induction hypothesis.

If u and v are neighbors, u is of level i and v of level $i+1$, then the edge (u,v) either appears in F_i and State(v) = Dual or it appears in K and State(v) = Even or Odd. Thus, if (v,u) is in F_i then if i is even then both vertices participate in G_{even} and if i is odd then both vertices participate in G_{odd}. By comparing the appropriate labels of u and v (either their $L(k-1, G_{even})$ label or their $L(k-1, G_{odd})$ label and by induction hypothesis we are able to deduce that u and v are indeed neighbors.

If State(v) = Even or Odd, then the edge (v,u) is in K so by looking at $L(v,K)$ in $L_k(G,v)$ and detecting $id(u)$ appearing there we conclude that u and v are indeed neighbors.

It is clear that if u and v are *not* neighbors in G then they are not neighbors in either one of the decomposed subgraphs, and therefore, by induction hypothesis we can never deduce that they are neighbors by our procedure.

The size of the label $L_k(G,u)$ is, by induction, at most $2^k \log n$ since by Lemma 3.4, the size of $L(v,K)$ is at most $k \log n$ and both sizes of $L_{k-1}(G_{odd},u)$ and $L_{k-1}(G_{even},u)$ are at most $2^{k-1}\log n$. ∎

We get the following corollary.

COROLLARY 3.3. *Let $\mathcal{G}_n$ be the family of n-vertex graphs, then $\mathcal{L}(k-\text{v-conn}, \mathcal{G}_n) \le 2^k \log n$.*

4 A lower bound for vertex connectivity on general graphs

In this section we establish a lower bound of $\Omega(k \log n)$ on the required label size for k-vertex connectivity on the class of n-vertex graphs where k is polylogarithmic in n. Fix a constant integer $c \ge 1$, assume that $k \le \log^c n$ and let $\mathcal{G}_m$ be the class of all $m = \frac{n}{2(k^2-k)}$-vertex graphs $\langle V,E\rangle$ with fixed id's $\{v_1,\ldots,v_m\}$ and degree at most $k-1$. Transform a given graph $G \in \mathcal{G}_m$ into a graph $T(G) = H$ with n vertices in the following way. Replace each edge $e_{i,j} = (v_i,v_j)$ in G by k vertices $w^1_{i,j}$ through $w^k_{i,j}$ and connect all the $w^l_{i,j}$'s to both v_i and v_j. Since G has at most $\frac{n}{2k}$ edges, H has at most n vertices. If necessary, add arbitrary isolated vertices to H so that it has precisely n vertices.

OBSERVATION 4.1. *Two vertices v_i, v_j are adjacent in G iff u and v are k-vertex-connected in $T(G) = H$.*

Assume we have a labeling scheme $\langle \mathcal{M}, \mathcal{D}\rangle$ for k-vertex connectivity on n-vertex graphs.

OBSERVATION 4.2. *Consider two distinct graphs $G_1, G_2 \in \mathcal{G}_m$, and let $L_i = \mathcal{M}(T(G_i))$ for $i = 1,2$. Then there exists a vertex v_j in V such that $L_1(v_j) \ne L_2(v_j)$, i.e., $\{L_1(v_1),\ldots,L_1(v_m)\} \ne \{L_2(v_1),\ldots,L_2(v_m)\}$.*

Since the number of graphs in $\mathcal{G}_m$ is $(\frac{m}{k})^{\Omega(km)}$ which is $m^{\Omega(km)}$ for k polylogarithmic in n, we get the following corollary.

COROLLARY 4.1. *There exists a graph $G \in \mathcal{G}(k)$ such that $\{L(v_1),\ldots,L(v_m)\}$ consists of at least $\log m^{\Omega(km)} = \Omega(km \log m)$ bits where $L = \mathcal{M}(G)$.*

We get the following theorem.

THEOREM 4.1. *$\mathcal{L}(k-\text{v-conn}, \mathcal{G}_n) = \Omega(k \log m) = \Omega(k \log n)$ for k polylogarithmic in n.*

References

[1] S. ABITEBOUL, H. KAPLAN, AND T. MILO, *Compact labeling schemes for ancestor queries*, in Proc. 12th ACM-SIAM Symp. on Discrete Algorithms, Jan. 2001.

[2] S. ALSTRUP, C. GAVOILLE, H. KAPLAN, AND T. RAUHE, *Identifying nearest common ancestors in a distributed environment*, Technical Report IT-C Series 2001-6, The IT University of Copenhagen, Aug. 2001.

[3] M. A. BREUER, *Coding the vertexes of a graph*, IEEE Trans. on Information Theory, IT-12 (1966), pp. 148–153.

[4] M. A. BREUER AND J. FOLKMAN, *An unexpected result on coding the vertices of a graph*, J. of Mathematical Analysis and Applications, 20 (1967), pp. 583–600.

[5] M. CHROBAK AND D. EPPSTEIN, *Planar orientations with low out-degree and compaction of adjacency matrices*, Theoret. Computer Sci., 86 (1991), pp. 243–266.

[6] S. EVEN, *Graph Algorithms*, Computer Science Press, 1979.

[7] C. GAVOILLE, M. KATZ, N. A. KATZ, C. PAUL, AND D. PELEG, *Approximate distance labeling schemes*, in Proc. 9th European Symp. on Algorithms, F. M. auf der Heide, ed., vol. 2161 of Lecture Notes in Computer Science, Springer, Aug. 2001, pp. 476–488.

[8] C. GAVOILLE AND C. PAUL, *Split decomposition and distance labelling: an optimal scheme for distance hereditary graphs*, in European Conf. on Combinatorics, Graph Theory and Applications, Sept. 2001.

[9] C. GAVOILLE AND D. PELEG, *Compact and localized distributed data structures*, Research Report RR-1261-01, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, Aug. 2001.

[10] C. GAVOILLE, D. PELEG, S. PÉRENNES, AND R. RAZ, *Distance labeling in graphs*, in Proc. 12th ACM-SIAM Symp. on Discrete Algorithms, ACM-SIAM, Jan. 2001, pp. 210–219.

[11] S. KANNAN, M. NAOR, AND S. RUDICH, *Implicit representation of graphs*, in Proc. 20th ACM Symp. on Theory of Computing, May 1988, pp. 334–343.

[12] H. KAPLAN AND T. MILO, *Parent and ancestor queries using a compact index*, in Proc. 20th ACM Symp. on Principles of Database Systems, May 2001.

[13] ———, *Short and simple labels for small distances and other functions*, in Proc. Workshop on Algorithms and Data Structures, Aug. 2001.

[14] M. KATZ, N. A. KATZ, A. KORMAN, AND D. PELEG, *Labeling schemes for flow and connectivity*, Technical Report MCS01-10, The Weizmann Institute of Science, 2001.

[15] M. KATZ, N. A. KATZ, AND D. PELEG, *Distance labeling schemes for well-separated graph classes*, in Proc. 17th Symp. on Theoretical Aspects of Computer Science, Feb. 2000, pp. 516–528.

[16] D. PELEG, *Proximity-preserving labeling schemes and their applications*, in Proc. 25th Int. Workshop on Graph-Theoretic Concepts in Computer Science, June 1999, pp. 30–41.

[17] ———, *Informative labeling schemes for graphs*, in Proc. 25th Symp. on Mathematical Foundations of Computer Science, vol. LNCS-1893, Springer-Verlag, Aug. 2000, pp. 579–588.