Labeling Schemes for Vertex Connectivity

AMOS KORMAN
CNRS and Université Paris Diderot - Paris 7, France.

This paper studies labeling schemes for the vertex connectivity function on general graphs. We consider the problem of assigning short labels to the nodes of any n-node graph is such a way that given the labels of any two nodes u and v, one can decide whether u and v are k-vertex connected in G, i.e., whether there exist k vertex disjoint paths connecting u and v. The paper establishes an upper bound of $k^2 \log n$ on the number of bits used in a label. The best previous upper bound for the label size of such a labeling scheme is $2^k \log n$.

Categories and Subject Descriptors: C.2.4 [**Computer-communication networks**]: Distributed Systems; E.1 [**Data Structures**]: Distributed data structures, Graphs and networks; G.2.1 [**Discrete mathematics**]: Combinatorics; G.2.2 [**Discrete mathematics**]: Graph theory

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Graph algorithms, vertex-connectivity, labeling schemes

1. INTRODUCTION

1.1 Problem and motivation

Network representations have played an extensive and often crucial role in many domains of computer science, ranging from data structures, graph algorithms to distributed computing and communication networks. Traditional network representations are usually global in nature; in order to derive useful information, one must access a global data structure representing the entire network, even if the sought information is local, pertaining to only a few nodes.

In contrast, the notion of *labeling schemes* (introduced in [Breuer 1966; Breuer and Folkman 1967; Kannan et al. 1992]) involves using a more *localized* representation of the network. 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 recently developed for different graph families and for a variety information types, including vertex adjacency [Alstrup

Author's address: Amos Korman: CNRS and Université Paris Diderot - Paris 7, France. E-mail: amos.korman@liafa.jussieu.fr. Supported in part by the project "GANG" of INRIA.

and Rauhe 2002; Breuer 1966; Breuer and Folkman 1967; Gavoille and Labourel 2007; Kannan et al. 1992; Korman et al. 2006], distance [Alstrup et al. 2005; Cohen et al. 2002; Gavoille et al. 2001; Gavoille and Labourel 2007; Gavoille et al. 2001; Katz et al. 2000; Kaplan and Milo 2001; Korman et al. 2006; Peleg 1999; Thorup 2004], tree routing [Fraigniaud and Gavoille 2001; Fraigniaud and Gavoille 2002; Thorup and Zwick 2001], vertex-connectivity [Alstrup and Rauhe 2002; Katz et al. 2004], flow [Korman and Kutten 2007; Katz et al. 2004], tree ancestry [Abiteboul at al. 2001; Abiteboul at al. 2006; Kaplan et al. 2002], nearest common ancestor in trees [Alstrup et al. 2004; Peleg 2000] and various other tree functions, such as center and separation level [Peleg 2000]. See [Gavoille and Peleg 2003] for a survey on static labeling schemes. The dynamic version was studied in [Cohen et al. 2002; Korman 2007; Korman 2008; Korman and Kutten 2007; Korman and Peleg 2007; Korman et al. 2004].

The current paper studies informative labeling schemes supporting the vertex connectivity function of general graphs. This type of information may be 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 number of available routes between any two given endpoints. In this paper, for any $k > 3$, we establish a labeling scheme supporting the k-vertex connectivity function on general n-node graphs using labels of size at most $\frac{k^2}{2} \log n$. The best previous upper bound for the label size of such a labeling scheme was shown in [Katz et al. 2004] to be $2^k \log n$.

1.2 Labeling schemes

Let f be a function on pairs of vertices. An *f-labeling scheme* $\pi = \langle \mathcal{M}, \mathcal{D} \rangle$ for the graph family $\mathcal{G}$ is composed of the following components:

(1) A *marker* algorithm $\mathcal{M}$ that given a graph in $\mathcal{G}$, assigns labels to its vertices.
(2) A polynomial time *decoder* algorithm $\mathcal{D}$ that given the labels $L(u)$ and $L(v)$ of two vertices u and v in some graph in $\mathcal{G}$, outputs $f(u, v)$.

It is important to note that the decoder $\mathcal{D}$, responsible of the f-computation, is independent of the graph G. Thus $\mathcal{D}$ can be viewed as a method for computing f-values in a "distributed" fashion, given any pair of labels and knowing that the graph belongs to some specific family $\mathcal{G}$. In contrast, the labels contain some information that can be precomputed by considering the whole graph structure. Therefore, in a sense, the area of labeling schemes can be viewed as "intermediate" between the sequential and distributed fields.

The common complexity measure used to evaluate a labeling scheme $\pi = \langle \mathcal{M}, \mathcal{D} \rangle$ is the *Label Size*, $\mathcal{L}_{\mathcal{M}}(\mathcal{G})$: the maximum number of bits in a label assigned by the marker algorithm $\mathcal{M}$ to any vertex in any graph 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 Vertex connectivity

Let $G = \langle V, E \rangle$ be an unweighted undirected graph. 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* $\texttt{v-conn}(u, w)$ of two vertices u and w in G equals the cardinality of the largest set P of vertex-disjoint paths connecting them. By Menger's theorem (cf. [Even 1979]), for non-adjacent u and w, $\texttt{v-conn}(u, w)$ 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.) The *k-vertex connectivity* function between nodes u and w, denoted $k - \texttt{v-conn}(u, w)$, is a boolean (TRUE/FALSE) function that is TRUE iff $\texttt{v-conn}(u, w) \geq k$, i.e., iff there exist at least k vertex-disjoint paths connecting u and w.

1.4 Related work and our contribution

Labeling schemes supporting the k-vertex connectivity function on general n-node graphs were previously studied in [Katz et al. 2004]. The label sizes achieved therein are $\lceil \log n \rceil$ for $k = 1$, $3\lceil \log n \rceil$ for $k = 2$, $5\lceil \log n \rceil$ for $k = 3$ and $2^k \lceil \log n \rceil$ for $k > 3$. Based on a counting argument, the authors also present a lower bound of $\Omega(k \log \frac{n}{k^3})$ for the required label size of such a labeling scheme.

In [Alstrup and Rauhe 2002] the authors establish an adjacency labeling scheme on the class of n-node graphs with arboricity[1] k using $k\lceil \log n \rceil + O(\log^* n)$-bit labels. Using their adjacency labeling scheme, the label sizes of the schemes in [Katz et al. 2004] for $k = 2$ and $k = 3$ can be reduced to $2 \log n + O(\log^* n)$ and $4 \log n + O(\log^* n)$, respectively.

In this paper, for any $k > 3$, we establish a k-vertex connectivity labeling scheme on general n-node graphs with label size at most $\frac{k^2}{2} \lceil \log n \rceil$.

Our labeling scheme is based on a novel decomposition of graphs closed under k-connectivity, i.e, graphs G which satisfy the property that any two k-vertex connected vertices in G, are neighbors. Specifically, we show that the edge set of any graph closed under k-connectivity can be partitioned into a collection of cliques and $\frac{k(k-1)}{2}$ forests.

2. PRELIMINARIES

In an undirected graph G, two vertices u and v are called *k-connected* if there exist at least k vertex-disjoint paths between them, i.e., if $\texttt{v-conn}(u, v) \geq k$. Given a graph $G = \langle V, E \rangle$ and two vertices u from v in V, a set $S \subseteq V$ *separates* u from v if u and v are not connected in the vertex induced subgraph $G \setminus S$.

THEOREM 2.1. **[Menger] (cf. [Even 1979])** *In an undirected graph G, two non-adjacent vertices u and v are k-connected iff no set $S \subset G \setminus \{u, v\}$ of $k - 1$ vertices separates 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 any two k-connected vertices in G are also neighbors in G. Let $\mathcal{C}(k)$ be the family of all graphs which are closed under k-connectivity.

Two subgraphs $H, F \in G$ are *vertex-disjoint subgraphs* if they share no common vertex. For graphs $G = \langle V, E \rangle$ and $G_i = \langle V, E_i \rangle$, $i > 1$, we say that G can be *edge-partitioned* into the G_i's if $\bigcup_i E_i = E$.

[1] A graph G has arboricity k if its edge set can be partitioned to at most k forests.

OBSERVATION 2.2. *A graph in $\mathcal{C}(1)$ is a collection of vertex-disjoint cliques.*

OBSERVATION 2.3. *(1) If $G \in \mathcal{C}(k)$ then each connected component of G belongs to $\mathcal{C}(k)$.*

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

The following two lemmas are taken from [Katz et al. 2004].

LEMMA 2.4. *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., $C_k(G) \in \mathcal{C}(k)$.*

LEMMA 2.5. *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., $C_k(G) = C_k(G')$.*

A graph G has *arboricity* k if the edges from the graph can be partitioned to at most k forests. Note that in particular the edges of a graph with arboricity k can be oriented such that the out-degree of each vertex is bounded from above by k. The class of graphs with arboricity k is denoted $\mathcal{A}(k)$, and the class on n-node graphs in $\mathcal{A}(k)$ is denoted $\mathcal{A}_n(k)$.

OBSERVATION 2.6. *If $G = H \sqcup F$ where $H, F \in \mathcal{A}(k)$ are vertex-disjoint subgraphs of G, then $G \in \mathcal{A}(k)$.*

A simple adjacency labeling scheme can be constructed for the graph family $\mathcal{A}_n(k)$, using $(k+1)\lceil \log n \rceil$-bit labels. This can be done as follows. Given an n-node graph $K \in \mathcal{A}_n(k)$, first assign a unique identity $id(v)$ to each vertex v, in the range $\{1, 2, \cdots n\}$, and then orient the edges in K so that the out-degree of each vertex is at most k. Now, label each vertex v by $k+1$ fields, such that the first field contains $id(v)$ and the remaining fields contain the identities of the nodes pointed by v. Given the labels of two vertices, in order to check adjacency, one simply checks whether the identity of one vertex appears in the list of identities in the label of the other vertex.

Let $\psi(k, n)$ denote the minimum number of bits needed for encoding adjacency in $\mathcal{A}_n(k)$, i.e., $\psi(k, n) = \mathcal{L}(adjacency, \mathcal{A}_n(k))$. Clearly, the above scheme indicates that $\psi(k, n) \leq (k+1)\lceil \log n \rceil$. Alstrup and Rauhe showed how to further reduce this bound.

THEOREM 2.7 CF. [ALSTRUP AND RAUHE 2002]. $\psi(k, n) = k\lceil \log n \rceil + O(\log^* n)$.

3. VERTEX-CONNECTIVITY LABELING SCHEMES FOR GENERAL GRAPHS

Let $\mathcal{G}_n$ denote the family of all undirected graphs with at most n vertices. In this section we present a k-vertex connectivity labeling scheme for the graph family $\mathcal{G}_n$ using less than $k^2 \log n$-bit labels. Let $\mathcal{C}_n(k) = \mathcal{C}(k) \cap \mathcal{G}_n$, i.e, $\mathcal{C}_n(k)$ is the family of all graphs with at most n vertices, which are closed under k-connectivity.

As in [Katz et al. 2004], we rely on the basic observation that labeling k-connectivity for some graph G is equivalent to labeling adjacency for $C_k(G)$. By Lemma 2.4, $C_k(G) \in \mathcal{C}_n(k)$ for every graph $G \in \mathcal{G}_n$. Therefore, instead of presenting a k-connectivity labeling scheme for $\mathcal{G}_n$, we present an adjacency labeling scheme for the graph family $\mathcal{C}_n(k)$.

The general idea used in [Katz et al. 2004] for labeling adjacency for some $G \in \mathcal{C}_n(k)$, is to partition the edges of G into a 'simple' graph in $\mathcal{A}_n(k)$ and two other graphs belonging to $\mathcal{C}_n(k-1)$. The labeling algorithm of [Katz et al. 2004] recursively labels subgraphs of G that belong to $\mathcal{C}_n(t)$ for $t < k$. Since adjacency for a graph in $\mathcal{A}_n(k)$ can be encoded using roughly $k \log n$-bit labels, the resulted labels in the recursive labeling use at most $2^k \log n$ bits. Our main technical contribution in this paper is to show that the edge set of any graph $G \in \mathcal{C}_n(k)$ can be partitioned into a 'simple' graph in $\mathcal{A}_n(k-1)$ and (only) one other graph belonging to $\mathcal{C}_n(k-1)$. Thus, using recursion, it follows that the edges of any graph $G \in \mathcal{C}_n(k)$ can be partitioned into a collection of cliques and $\frac{k(k-1)}{2}$ forests, i.e., into one graph in $\mathcal{C}_n(1)$ and one graph which has arboricity $\frac{k(k-1)}{2}$.

The labeling scheme of size $\psi(\frac{k(k-1)}{2}, n) + \lceil \log n \rceil$ (which is at most $\frac{k^2}{2}\lceil \log n \rceil$ if $k > 3$) follows by separately labeling adjacency for each of the two decomposed graphs.

3.1 The decomposition

We now show that any graph $G \in \mathcal{C}_n(k)$ can be edge-partitioned into one 'simple' graph in $\mathcal{A}_n(k-1)$ and (only) one other graph belonging to $\mathcal{C}_n(k-1)$. Consider a graph $G \in \mathcal{C}_n(k)$, and let $C_1, ..., C_m$ be its connected components. Fix i and let $C = C_i$ be one of these connected components. By the first part in Observation 2.3, $C \in \mathcal{C}(k)$.

Let $T \equiv T(C)$ denote a shortest path tree spanning C rooted at some vertex r. For a vertex $v \in C$, let $depth(v)$ denote its depth in T, i.e, its (hop) distance in T to the root r. An *uncle* of a vertex $v \in C$ is a neighbor of v (in the graph C) at depth $depth(v) - 1$ (in particular, the parent of v in T is an uncle of v). For every vertex $v \in C$, let $Deg_{up}(v)$ denote the number of v's uncles. For a vertex $v \in C$, let $Deg^*_{up}(v) = \min(Deg_{up}(v), k-1)$.

We now define a subgraph of C called K which has arboricity $k-1$. For each vertex $v \in C$, let $U(v)$ be some set containing $Deg^*_{up}(v)$ uncles of v. The graph K is the graph obtained by the set of edges $\{(v,u) \mid v \in C \text{ and } u \in U(v)\}$. Clearly, K has arboricity $k-1$. Let H be the graph obtained from C by removing the edges of K, i.e., $H = C \setminus K$. Note that H may not be connected. See Figure 1.

At this point, we note that a desired decomposition of C would have been obtained had H been in $\mathcal{C}(k-1)$. However, this may not necessarily be the case. Instead, we proceed as follows. We show that there exists a graph H' such that $H \subseteq H' \subseteq C$ and H' is in $\mathcal{C}(k-1)$. Therefore, in particular, $K' = C \setminus H'$ is a subgraph of K and therefore also has arboricity $k-1$. The desired decomposition of C is thus given by K' and H'.

Before we proceed, we need the following definition. The *closure* of H, denoted $Close(H)$, is the graph obtained by adding the edges of $C_{k-1}(H)$ to H (i.e., $Close(H) = H \cup C_{k-1}(H)$).

LEMMA 3.1. *$Close(H) \in \mathcal{C}_n(k-1)$.*

Proof: Consider the process of constructing $Close(H)$ by adding the edges of $C_{k-1}(H)$ to H, one by one. The lemma follows by induction on the steps of this process using Lemma 2.5. ∎

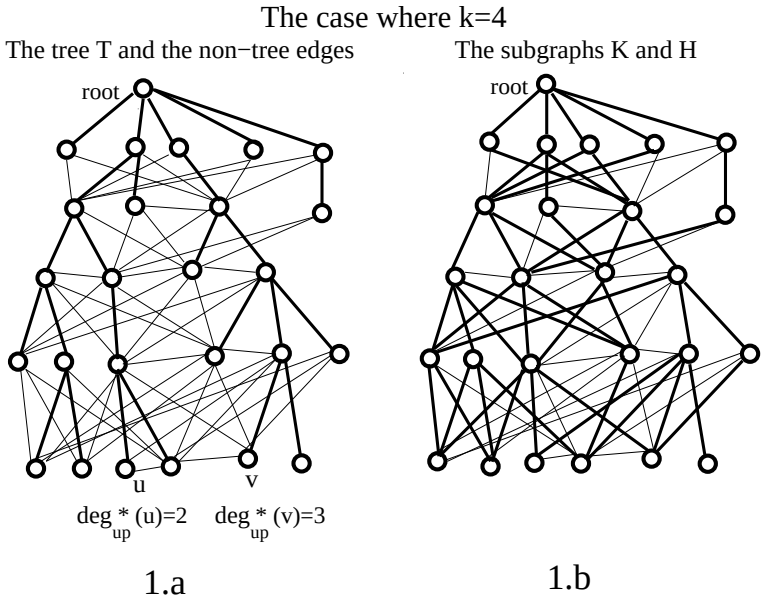

Fig. 1. Both figures represent the graph C. Consider the case were $k = 4$. In the left figure, 1.a, the thick lines represented the edges of the shortest path tree T, and the thin lines represent the non-tree edges of C. Note that $deg_{up}(u) = deg^*_{up}(u) = 2$, $deg_{up}(v) = 4$ and $deg^*_{up}(v) = 3$. In figure 1.b, the thick lines represented the edges of K, and the thin lines represent the edges of $H = C \setminus K$.

□

LEMMA 3.2. *$Close(H)$ is a subgraph of C.*

Proof: In order to show that $Close(H) \subseteq C$, it is enough to show that $C_{k-1}(H) \subseteq C$, i.e., for any pair of vertices u and v, if u and v are $k-1$ connected in H then they are neighbors in C.

Let u and v be two vertices which are $k-1$ connected in H. Assume, by contradiction, that u and v are not neighbors in C. Since C is closed under k-connectivity, then u and v are also not k-connected in C. Therefore, by Menger's theorem, since u and v are non-adjacent in C, there exist a set $S \subset C \setminus \{u, v\}$ of $k-1$ vertices which separates u from v in C. In particular, S separates u from v in H. Let $s \in S$ be a vertex of lowest depth among the vertices in S, and let $S' = S \setminus \{s\}$. See Figure 2.a.

Since u and v are $k-1$ connected in H, it follows by Menger's theorem that there is no strict subset of S which separates u from v in H. In particular, there exists a path P in $H \setminus S'$ which connects u with v. Note that s must belong to P. See Figure 2.b.

We now show that in C, the set S does not separate u from the root r of C. First note, that in the case where $depth(s) \geq depth(u)$, then clearly, S does not separate u from r even in T. Consider now the case where $depth(s) < depth(u)$. Let w be the last vertex on the subpath of P connecting u and s such that $depth(w) = depth(s)+1$ (possibly w is u itself). Note that after w, the next vertex w' on the subpath of

P connecting u and s, has depth $depth(w') = depth(s)$ (possibly w' is s itself). Observe that if $w' \neq s$ then both the subpath of P connecting u and w' and the shortest path on T connecting w' and r belong to $C \setminus S$. Thus, we get that if $w' \neq s$ then S does not separate u from r in C. The more interesting case is when $w' = s$. See Figure 2.b. In this case, since the edge (w, w') belongs to H, it follows that the out-degree of w in K is $k-1$. Thus, w has at least k uncles in C. See Figure 2.c. In particular, w has an uncle not in S. Moreover, since there is no vertex in S of smaller depth than s, we obtain that there is a path connecting w and r in $C \setminus S$. Since the subpath of P connecting u and w also belongs to $C \setminus S$, we obtain that in C, the set S does not separate u from r. Similarly, v is connected to r in $C \setminus S$. See Figure 2.d. It therefore follows that in C, the set S does not separate u from v. Contradiction. ▮

□

LEMMA 3.3. *C can be edge-partitioned into a graph in $\mathcal{C}(k-1)$ and a graph in $\mathcal{A}(k-1)$.*

Proof: By Lemma 3.2, the subgraph $K' = C \setminus Close(H)$ is well defined. Since $H \subseteq Close(H)$, we obtain that $K' \subseteq K$. Therefore, since K has arboricity $k-1$, then so does K'. By Lemma 3.1, $Close(H) \in \mathcal{C}(k-1)$. It therefore follows that C can be edge-partitioned into the graph $Close(H) \in \mathcal{C}(k-1)$ and the graph $K' \in \mathcal{A}(k-1)$. ▮

□

Using Observations 2.3 and 2.6, we obtain the following corollary.

COROLLARY 3.4. *Each $G \in \mathcal{C}_n(k)$ can be edge-partitioned into a graph in $\mathcal{C}_n(k-1)$ and a graph in $\mathcal{A}_n(k-1)$.*

Using induction, we get the following.

THEOREM 3.5. *Each $G \in \mathcal{C}_n(k)$ can be edge-partitioned into a graph $G_1 \in \mathcal{C}_n(1)$ (which is a collection of cliques) and one graph $G_2 \in \mathcal{A}_n(\frac{k(k-1)}{2})$.*

3.2 The labeling scheme

THEOREM 3.6. *For any $k > 1$, we have $\mathcal{L}(\texttt{adjacency}, \mathcal{C}_n(k)) \leq \psi(\frac{k(k-1)}{2}, n) + \lceil \log n \rceil$.*

Proof: We describe an adjacency labeling scheme $\pi = \langle \mathcal{M}, \mathcal{D} \rangle$ for $\mathcal{C}_n(k)$. Given a graph $G \in \mathcal{C}_n(k)$, we partition the edges of G according to Theorem 3.5. Note first, that since a graph in $\mathcal{C}_n(1)$ is simply a collection of cliques, $\mathcal{L}(\texttt{adjacency}, \mathcal{C}_n(1)) \leq \lceil \log n \rceil$ (in fact, since disjoint labels are not necessary here, $\lceil \log t \rceil$ bits suffice, where t is the number of connected components). In other words, there exists an adjacency labeling scheme $\pi_1 = \langle \mathcal{M}_1, \mathcal{D}_1 \rangle$ for $\mathcal{C}_n(1)$ of size $\lceil \log n \rceil$. By padding enough zeros to the left of these labels, we may assume that each label in π_1 is composed of precisely $\lceil \log n \rceil$ bits. For a vertex $u \in G_1$, let $L_1(u)$ denote the label given to u by π_1. Let $\pi_2 = \langle \mathcal{M}_2, \mathcal{D}_2 \rangle$ be an adjacency labeling scheme for $\mathcal{A}_n(\frac{k(k-1)}{2})$ with label size $\psi(\frac{k(k-1)}{2}, n)$. For each vertex $u \in G_2$, let $L_2(u)$ denote the label given to u by π_2.

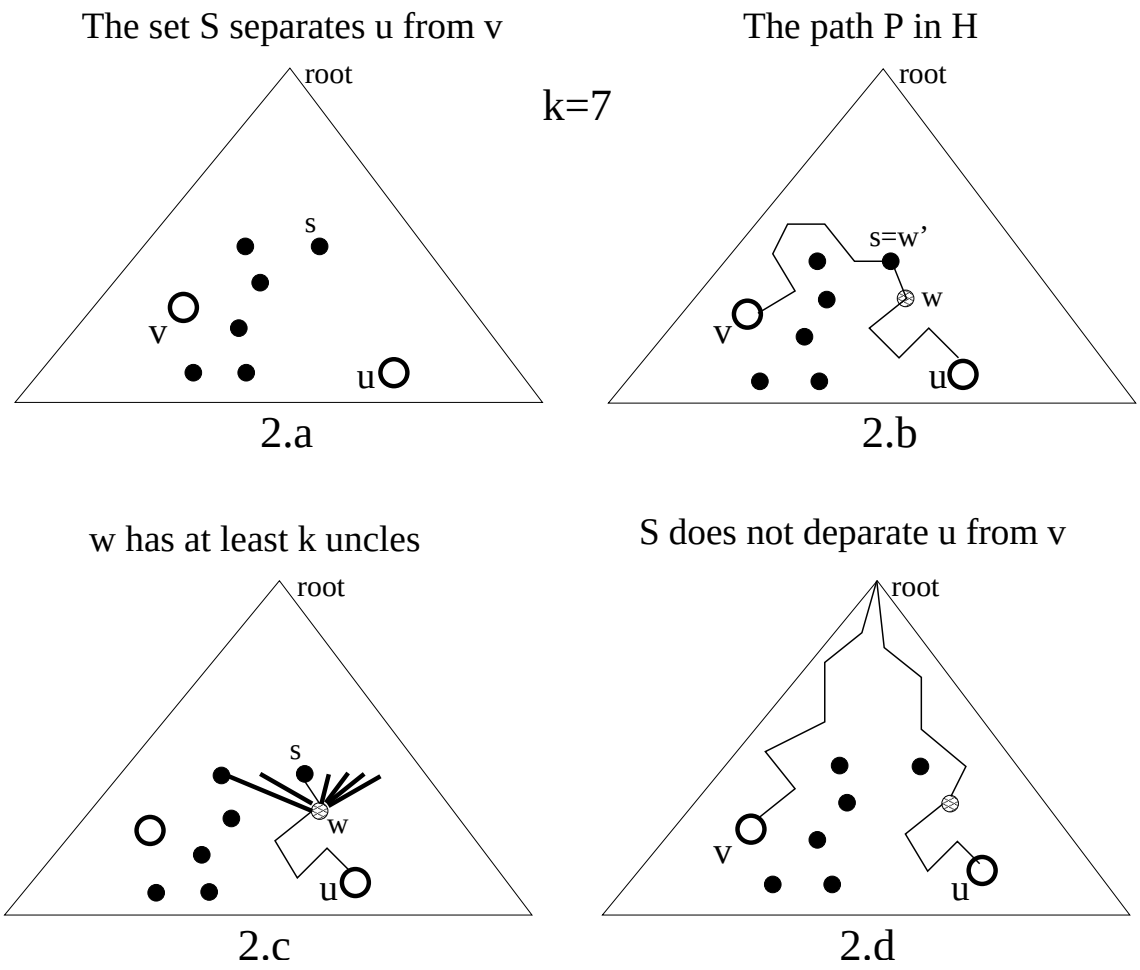

Fig. 2. The figures consider the case $k = 7$. In Figure 2.a, the black circles represent the vertices of S, where s is the vertex of lowest depth among them. In Figure 2.b, the path represents the path P in H which connects u and v and passes through s. Vertex w is the last vertex on the subpath of P from u to s with depth $depth(w) = depth(s) + 1$, and we examine the case where $w' = s$. In Figure 2.c, the thick lines represent the outgoing edges of w in K, connecting w with $k - 1$ uncles. Therefore, w has at least k uncles in C. In Figure 2.d, the paths connect u and v with the root, without passing through any vertex in S.

For each vertex $u \in G$, the label given to u by the marker $\mathcal{M}$ is $L(u) = \langle L_1(u), L_2(u) \rangle$. We therefore get that the label size of π is $\psi(\frac{k(k-1)}{2}, n) + \lceil \log n \rceil$. We would like to point out that if the labeling scheme π_2 assigns disjoint labels then clearly $\mathcal{M}$ also assigns disjoint label. The previously mentioned adjacency schemes for $\mathcal{A}_n(\frac{k(k-1)}{2})$ assign disjoint labels. Thus, if we use one of these schemes instead of π_2 then the marker $\mathcal{M}$ will assign disjoint labels as well.

Given the labels $L(u)$ and $L(v)$ of two vertices u and v in some $G \in \mathcal{C}(k)$, the decoder $\mathcal{D}$ outputs 1 iff either $\mathcal{D}_1(L_1(u), L_1(v)) = 1$ or $\mathcal{D}_2(L_2(u), L_2(v)) = 1$. Note that the decoder $\mathcal{D}$ can distinguish between the two fields in a label, since the

number of bits in the first field is fixed in advance. Recall also that in the definition of a labeling scheme we require the decoder to be polynomial in the size of the labels. The decoder $\mathcal{D}$ is indeed polynomial since $\mathcal{D}_1$ operates in constant time and $\mathcal{D}_2$ is polynomial as π_2 is a labeling scheme.

The fact that the labeling scheme π is a correct adjacency labeling scheme for $\mathcal{C}_n(k)$ follows from Theorem 3.5 and from the correctness of the adjacency labeling schemes π_1 and π_2. ▮

□

The following corollary matches the best known bound for labeling k-vertex connectivity on n-nodes graphs, in the cases $k = 2$ and $k = 3$, and significantly improves the bound for general $k > 3$.

COROLLARY 3.7. *(1) For any $k > 1$, $\mathcal{L}(k-\texttt{v-conn}, \mathcal{G}_n) \leq \psi(\frac{k(k-1)}{2}, n)+\lceil \log n \rceil$. (2) For any $k > 3$, $\mathcal{L}(k - \texttt{v-conn}, \mathcal{G}_n) \leq \frac{k^2}{2}\lceil \log n \rceil$.*

REFERENCES

ABITEBOUL, S., AND KAPLAN, H., AND MILO, T. 2001. Compact labeling schemes for ancestor queries. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, 547–556.

ABITEBOUL, S., AND ALSTRUP, S., AND KAPLAN, H., AND MILO, T., AND RAUHE, T. 2006. Compact Labeling Scheme for Ancestor Queries. *SIAM Journal on Computing* 35, 1295–1309.

ALSTRUP, S., AND BILLE, P., AND RAUHE, T. 2005. Labeling schemes for small distances in trees. *SIAM Journal of Discrete Math* 19, 448–462.

ALSTRUP, S., AND GAVOILLE, C., AND KAPLAN, H., AND RAUHE, T. 2004. Nearest Common Ancestors: A Survey and a new Distributed Algorithm. *Theory of Computing Systems* 37, 441–456.

ALSTRUP, S., AND RAUHE, T. 2002. Small induced-universal graphs and compact implicit graph representations. In *Proc. 43'rd annual IEEE Symp. on Foundations of Computer Science*, 53–62.

BREUER, M., A. 1966. Coding the vertexes of a graph. *IEEE Transactions on Information Theory*, IT-12:148–153, 1966.

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

COHEN, E., AND HALPERIN, E., AND KAPLAN, H., AND ZWICK, U. 2002. Reachability and Distance Queries via 2-hop Labels. In *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms*, 937–946.

COHEN, E., AND KAPLAN, H., AND MILO, T. 2002. Labeling dynamic XML trees. In *Proc. 21st ACM Symp. on Principles of Database Systems*, 271–281.

FRAIGNIAUD, P., AND GAVOILLE, C. 2001. Routing in trees. In *Proc. 28th Int. Colloq. on Automata, Languages & Prog.*, 757–772.

FRAIGNIAUD, P., AND GAVOILLE, C. 2002. A space lower bound for routing in trees. In *Proc.* 19^{th} *Symp. on Theoretical Aspects of Computer Science*, 65–75.

GAVOILLE, C., AND KATZ, M., AND KATZ, N., A., AND PAUL, C., AND PELEG, D. 2001. Approximate Distance Labeling Schemes. In *9th Ann. European Symp. on Algorithms*, 476–488.

GAVOILLE, C., AND LABOUREL, A. 2007. Shorter Implicit Representation for Planar Graphs and Bounded Treewidth Graphs. In *15th Ann. European Symp. on Algorithms*, 582–593.

GAVOILLE, C., AND LABOUREL, A. 2007. On local representation of distances in trees. In *Proc. 26th Ann. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, 352–353.

GAVOILLE, C., AND PELEG, D. 2003. Compact and Localized Distributed Data Structures. *Journal of Distributed Computing* 16, 111–120.

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

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

KATZ, M., AND KATZ, N., A., AND KORMAN, A., AND PELEG, D. 2004. Labeling schemes for flow and connectivity. *SIAM Journal on Computing* 34, 23–40.

KAPLAN, H., AND MILO, T. 2001. Short and simple labels for small distances and other functions. In *Workshop on Algorithms and Data Structures*, 246–257.

KAPLAN, H., AND MILO, T. AND SHABO, R. 2002. A Comparison of Labeling Schemes for Ancestor Queries. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms*, 954–963.

KANNAN, S., AND NAOR, M., AND RUDICH S. 1992. Implicit representation of graphs. *SIAM Journal on Descrete Math* 5, 596–603.

KORMAN, A. 2007. General Compact Labeling Schemes for Dynamic Trees. *Journal of Distributed Computing* 20(3), 179–193.

KORMAN, A. 2008. Improved compact routing schemes for dynamic trees. In *Proc. 27th Ann. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, 185–194.

KORMAN, A., AND KUTTEN, S. 2007. Distributed Verification of Minimum Spanning Trees. *Journal of Distributed Computing* 20(4), 253–266.

KORMAN, A., AND KUTTEN, S. 2007. Controller and Estimator for Dynamic Networks. In *Proc. 26th Ann. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, 175–184.

KORMAN, A., AND PELEG, D. 2007. Labeling Schemes for Weighted Dynamic Trees. *Journal of Information and Computation*. 205(12), 1721–1740.

KORMAN, A., AND PELEG, D., AND RODEH, Y. 2004. Labeling schemes for dynamic tree networks. *Theory of Computing Systems* 37, 49–75.

KORMAN, A., AND PELEG, D., AND RODEH, Y. 2006. Constructing Labeling Schemes through Universal Matrices. *Proc. 17th Int. Symposium on Algorithms and Computation.*, 409–418.

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

PELEG, D. 2000. Informative labeling schemes for graphs. In *Proc. 25th Symp. on Mathematical Foundations of Computer Science*, 579–588.

THORUP, M. 2004. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM* **51**, 993–1024.

THORUP, M. AND ZWICK, U. 2001. Compact routing schemes. In *Proc. 13th ACM Symp. on Parallel Algorithms and Architecture*, 1–10.

EVEN, S. 1979. *Graph Algorithms*. Computer Science Press.