Labeling Schemes for Vertex Connectivity

Amos Korman *

Abstract

This paper studies labeling schemes for the vertex connectivity function on general graphs. We consider the problem of labeling the nodes of any n-node graph is such a way that given the labels of 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 labeling scheme is $2^k \log n$.

Keywords: Graph algorithms, vertex-connectivity, labeling schemes.

*Contact person. Information Systems Group, Faculty of IE&M, The Technion, Haifa, 32000 Israel. E-mail:pandit@tx.technion.ac.il. Supported in part at the Technion by an Aly Kaufman fellowship.

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 [6, 7, 18]) 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 [6, 7, 18, 5, 26], distance [27, 23, 15, 14, 12, 19, 29, 8, 1, 26], tree routing [10, 11, 30], vertex-connectivity [22, 5], flow [17, 22], tree ancestry [3, 4, 21], nearest common ancestor in trees [2] and various other tree functions, such as center and separation level [28]. See [13] for a survey on static labeling schemes. The dynamic version was studied in [25, 24, 9, 16].

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. We establish a labeling scheme supporting the k-vertex connectivity function of general n-node graphs using $k^2 \log n$-bit labels. The best previous upper bound for the label size of such a labeling scheme was shown in [22] 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 G or of the number of vertices in it. Thus $\mathcal{D}$ 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 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 considered as intermediate between the sequential and distributed fields.

The most commonly 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. [31]), 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.)

1.4 Related work and our contribution

Labeling schemes supporting the k-vertex connectivity function on general n-node graphs were previously studied in [22]. The label sizes achieved therein were $\log n$ for $k = 1$, $3 \log n$ for $k = 2$, $5 \log n$ for $k = 3$ and $2^k \log n$ for $k > 3$. Based on a counting argument, they also present a lower bound of $\Omega(k \log \frac{n}{k^3})$ for the required label size of such a labeling scheme.

In [5] they establish an adjacency labeling scheme on n-node trees using $\log n + O(\log^* n)$-bit labels. Using this adjacency labeling scheme they show how to reduce the upper bounds on the label size of k-vertex connectivity labeling schemes to $2 \log n + O(\log^* n)$ in the the case $k = 2$ and to $4 \log n + O(\log^* n)$ in the the case $k = 3$.

In this paper we establish a k-vertex connectivity labeling scheme on general n-node graphs using $k^2 \log n$-bit labels.

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. [31]) *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 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 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 $G_i = \langle V_i, E_i\rangle$, $i > 1$, we say that G can be *decomposed* into the G_i's if $\bigcup_i V_i = V$ and $\bigcup_i E_i = E$.

Observation 2.2 *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 $G \in \mathcal{C}(k)$.

The following two lemmas are taken from [22].

Lemma 2.3 *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 2.4 *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 is called *k-orientable* if there exists an orientation of the edges such that the out-degree of each vertex is bounded from above by k. The class of k-orientable graphs is denoted $\mathcal{J}_{or}(k)$.

Observation 2.5 *If $G = H \cup F$ where $H, F \in \mathcal{J}_{or}(k)$ are vertex-disjoint subgraphs of G, then $G \in \mathcal{J}_{or}(k)$.*

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 $k^2 \log n$-bit labels. Let $\mathcal{C}_n(k) = \mathcal{C}(k) \bigcap \mathcal{G}_n$, 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 [22], 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 2.3, $C_k(G) \in \mathcal{C}_n(k)$ for every graph $G \in \mathcal{G}_n$. Therefore, instead of presenting a k-connectivity labeling scheme $\mathcal{G}_n$, we present an adjacency labeling scheme for the graph family $\mathcal{C}_n(k)$.

The general idea used in [22] for labeling adjacency for some $G \in \mathcal{C}_n(k)$, is to decompose G into a 'simple' k-orientable graph in $\mathcal{G}_n$ and two other graphs belonging to $\mathcal{C}_n(k-1)$. The labeling algorithm of [22] recursively labels subgraphs of G that belong to $\mathcal{C}_n(t)$ for $t < k$. Since adjacency in a k-orientable graph in $\mathcal{G}_n$ can be encoded using $k \log n$-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 any graph $G \in \mathcal{C}_n(k)$ can be decomposed into a 'simple' k-orientable graph and (only) one other graph belonging to $\mathcal{C}_n(k-1)$. Thus, using a recursive labeling we obtain our $k^2 \log n$ upper bound.

3.1 The decomposition

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 2.2, $C \in \mathcal{C}(k)$.

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

We now define a k-orientable subgraph of C called K. 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 is a k-orientable graph. 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.

Lemma 3.1 *If u and v are $k-1$ connected in H then they are neighbors in C.*

Proof: 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 $C \in \mathcal{C}(k)$, 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 hight among the vertices in S, and let $S' = S \setminus \{s\}$. Note that since u and v are $k-1$ connected in H, 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.

We now show that in C, the set S does not separate u from the root r of C. If s is not an ancestor of u in T then clearly, S does not separate u from r even in T. Otherwise, if s is an ancestor of u in T, let w be the child of s in the path P. Note that w is also an ancestor of u (possibly w is u itself). Since the edge (w, s) belongs to H, it follows that w has at least $k+1$ uncles in C. Therefore, w has an uncle not in S. Moreover, since there is no vertex in S of smaller hight 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 $C \setminus \{S\}$. It therefore follows that in C, the set S does not separate u from v. Contradiction. ∎

Lemma 3.2 *C can be decomposed into a graph in $\mathcal{C}(k-1)$ and a $(k-1)$-orientable graph.*

Proof: Transform the graph H into $\widehat{H} = H \cup C_{k-1}(H)$ by adding the edges of $C_{k-1}(H)$ to H, one by one. By induction on the steps of this process using Lemma 2.4, we get $C_{k-1}(\widehat{H}) = C_{k-1}(H) \subseteq \widehat{H}$. Therefore, $\widehat{H} \in \mathcal{C}(k-1)$. By Lemma 3.1, $\widehat{H}$ is a subgraph of C. It therefore follows that C can be decomposed to the graph $\widehat{H} \in \mathcal{C}(k-1)$ and the k-orientable graph K.

Using Observations 2.2 and 2.5, we obtain the following corollary.

Corollary 3.3 *Each $G \in \mathcal{C}_n(k)$ can be decomposed into a graph in $\mathcal{C}_n(k-1)$ and a $(k-1)$-orientable graph in $\mathcal{G}_n$.*

Using induction, we get the following corollary.

Corollary 3.4 *Each $G \in \mathcal{C}_n(k)$ can be decomposed into a graph $G_1 \in \mathcal{C}_n(1)$ and $k-1$ graphs $G_2, G_3, \cdots, G_k$ such that for each $2 \leq i \leq k$, the graph G_i is an $(i-1)$-orientable graph in $\mathcal{G}_n$.*

3.2 The labeling scheme

We begin with the following simple observation.

Observation 3.5 *Let $\mathcal{J}_n(k) = \mathcal{J}_{or}(k) \bigcap \mathcal{G}_n$ be the family of k-orientable graphs with at most n vertices. Then $\mathcal{L}(\texttt{adjacency}, \mathcal{J}_n(k)) \leq (k+1)\lceil \log n \rceil$.*

Proof: Given a graph $K \in \mathcal{J}_n(k)$, we first assign a unique identifier $id(u)$ in the range $\{1, 2, \cdots, n\}$ to each node $u \in K$. We may assume that $id(u)$ is encoded using precisely $\lceil \log n \rceil$ bits. (If $id(u)$ is encoded using less than $\lceil \log n \rceil$ bits, simply pad enough zeros to the left of the encoding.) We now orient the edges of the k-orientable graph K such that each node $u \in K$, points to at most k nodes. For each node $u \in K$, let $N(u)$ denote the set of nodes pointed by u. We have $|N(u)| \leq k$.

The label $L_{or}(u)$ assigned to each node u consists of $|N(u)| + 1$ fields. Each field contains exactly $\lceil \log n \rceil$ bits. The first field contains $id(u)$, and the other fields contain the identifiers of the nodes in $N(u)$. Given the labels $L_{or}(u)$ and $L_{or}(v)$ of two vertices u and v, the decoder outputs 1 iff either the first field in $L_{or}(u)$ appears as a field in $L_{or}(v)$ or the first field in $L_{or}(v)$ appears as a field in $L_{or}(u)$. Clearly, this is a correct adjacency labeling scheme for $\mathcal{J}_n(k)$ with label size at most $(k+1)\lceil \log n \rceil$. ∎

Theorem 3.6 $\mathcal{L}(\texttt{adjacency}, \mathcal{C}_n(k)) \leq k^2 \log n$.

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 decompose G according to Corollary 3.4. Note 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 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 $\lceil \log n \rceil$. For a vertex $u \in G_1$, let $L_1(u)$ denote the label given to u by π_1. By Observation 3.5, there exists an adjacency labeling scheme $\pi_i = \langle \mathcal{M}_i, \mathcal{D}_i \rangle$ for $\mathcal{J}_n(i)$ with label size $(i+1)\lceil \log n \rceil$. For each $2 \leq i \leq k$ and each vertex $u \in G_i$, let $L_i(u)$ denote the label given to u by π_{i-1}. As before, we may assume that for each $1 \leq i \leq k$ and each vertex $u \in G$, the label $L_i(u)$ is encoded using precisely $i\lceil \log n \rceil$ bits.

For each node $u \in G$, the label $L(u)$ given by $\mathcal{M}$ is $L(u) = \langle L_1(u), L_2(u), \cdots, L_k(u) \rangle$. 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 there exists $1 \leq i \leq k$ such that $\mathcal{D}_i(L_i(u), L_i(v)) = 1$.

The fact that the labeling scheme π is a correct adjacency labeling scheme for $\mathcal{C}_n(k)$ follows from Corollary 3.4 and from the correctness of the adjacency labeling schemes π_i. The fact that the label size of π is at most $\frac{k(k+1)}{2} \cdot \lceil \log n \rceil$ follows from Observation 3.5. If $k > 3$, then $\frac{k(k+1)}{2} \cdot \lceil \log n \rceil \leq k^2 \log n$. Therefore, the theorem follows by combining this inequality with the results of [22] for the cases $k = 1, 2$ and 3. ∎

Corollary 3.7 $\mathcal{L}(k - \texttt{v-conn}, \mathcal{G}_n) \leq k^2 \log n$.

References

[1] S. Alstrup, P. Bille and T. Rauhe. Labeling schemes for small distances in trees. In *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2003.

[2] S. Alstrup, C. Gavoille, H. Kaplan and T. Rauhe. Nearest Common Ancestors: A Survey and a new Distributed Algorithm. *Theory of Computing Systems* **37**, (2004), 441–456.

[3] S. Abiteboul, H. Kaplan and T. Milo. Compact labeling schemes for ancestor queries. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2001.

[4] S. Alstrup and T. Rauhe. Improved Labeling Scheme for Ancestor Queries. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002.

[5] S. Alstrup and T. Rauhe. Small induced-universal graphs and compact implicit graph representations. In *Proc. 43'rd annual IEEE Symp. on Foundations of Computer Science*, Nov. 2002.

[6] M.A. Breuer. Coding the vertexes of a graph. *IEEE Trans. on Information Theory*, IT-12:148–153, 1966.

[7] M.A. Breuer and J. Folkman. An unexpected result on coding the vertices of a graph. *J. of Mathematical Analysis and Applications* **20**, (1967), 583–600.

[8] E. Cohen, E. Halperin, H. Kaplan and U. Zwick. Reachability and Distance Queries via 2-hop Labels. In *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002.

[9] E. Cohen, H. Kaplan and T. Milo. Labeling dynamic XML trees. In *Proc. 21st ACM Symp. on Principles of Database Systems*, June 2002.

[10] P. Fraigniaud and C. Gavoille. Routing in trees. In *Proc. 28th Int. Colloq. on Automata, Languages & Prog.*, LNCS 2076, pages 757–772, July 2001.

[11] P. Fraigniaud and C. Gavoille. A space lower bound for routing in trees. In *Proc.* 19^{th} *Symp. on Theoretical Aspects of Computer Science*, Mar. 2002.

[12] C. Gavoille and C. Paul. Split decomposition and distance labelling: an optimal scheme for distance hereditary graphs. In *Proc. European Conf. on Combinatorics, Graph Theory and Applications*, Sept. 2001.

[13] C. Gavoille and D. Peleg. Compact and Localized Distributed Data Structures. *J. of Distributed Computing* **16**, (2003), 111–120.

[14] C. Gavoille, M. Katz, N.A. Katz, C. Paul and D. Peleg. Approximate Distance Labeling Schemes. In *9th European Symp. on Algorithms*, Aug. 2001, Aarhus, Denmark, SV-LNCS 2161, 476–488.

[15] C. Gavoille, D. Peleg, S. Pérennes and R. Raz. Distance labeling in graphs. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, pages 210–219, Jan. 2001.

[16] A. Korman. General Compact Labeling Schemes for Dynamic Trees. In *Proc. 19th International Symp. on Distributed Computing*, Sep. 2005.

[17] A. Korman and S. Kutten. Distributed Verification of Minimum Spanning Trees. *Proc. 25th Annual Symposium on Principles of Distributed Computing*, July 2006.

[18] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. In *SIAM J. on Descrete Math* **5**, (1992), 596–603.

[19] H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In *Workshop on Algorithms and Data Structures*, Aug. 2001.

[20] 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.

[21] H. Kaplan, T. Milo and R. Shabo. A Comparison of Labeling Schemes for Ancestor Queries. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2002.

[22] M. Katz, N.A. Katz, A. Korman and D. Peleg. Labeling schemes for flow and connectivity. *SIAM Journal on Computing* **34** (2004),23–40.

[23] 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*, pages 516–528, Feb. 2000.

[24] A. Korman and D. Peleg. Labeling Schemes for Weighted Dynamic Trees. In *Proc. 30th Int. Colloq. on Automata, Languages & Prog.*, Eindhoven, The Netherlands, July 2003, SV LNCS.

[25] A. Korman, D. Peleg and Y. Rodeh. Labeling schemes for dynamic tree networks. *Theory of Computing Systems* **37**, (2004), 49–75.

[26] A. Korman, D. Peleg and Y. Rodeh. Constructing Labeling Schemes through Universal Matrices. *Proc. 17th Int. Symposium on Algorithms and Computation. (ISAAC)*, Dec. 2006.

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

[28] D. Peleg. Informative labeling schemes for graphs. In *Proc. 25th Symp. on Mathematical Foundations of Computer Science*, volume LNCS-1893, pages 579–588. SV, Aug. 2000.

[29] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. of the ACM* **51**, (2004), 993–1024.

[30] M. Thorup and U. Zwick. Compact routing schemes. In *Proc. 13th ACM Symp. on Parallel Algorithms and Architecture*, pages 1–10, Hersonissos, Crete, Greece, July 2001.

[31] Shimon Even. *Graph Algorithms.* Computer Science Press, 1979.